

Wireless Mousetrap Monitoring System

ENTRY F2032

One of the things I learnt from my everyday engineering practice is that there is always room for ingenuity and improvements. This is very appropriate for this project, which applies a couple of inexpensive microcontrollers to a very unusual field: “live-catch” mousetraps.

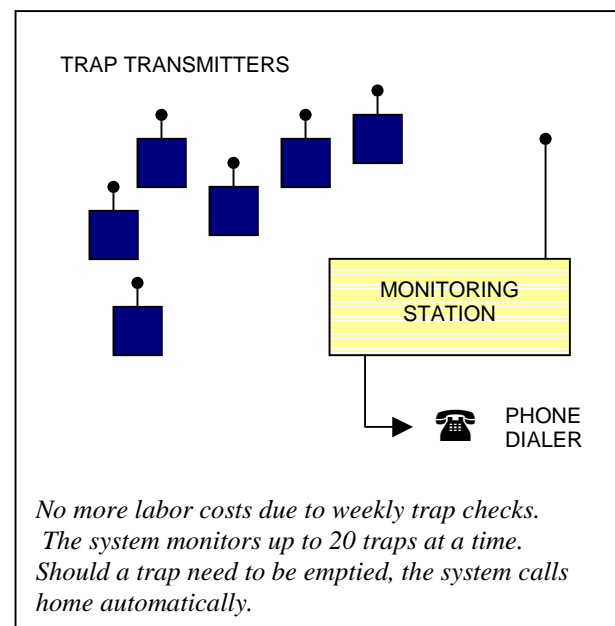
Mousetraps of this kind work imprisoning the mouse instead of killing him with chemicals or bloody mechanisms. They find large use when regulations, laws, or just plain common sense forbids poisons and chemicals. This is true for all of the stages in the food industry, starting from farms, to food conservation, logistics, processing, packaging, wholesale distributors and local retailers, down to your preferred restaurant. As you see, we are talking about a *huge* market with millions of customers, literally in any country of the world.

Other fields worth to mention are schools and public places where traps could be opened by children or pets; and even hospitals and pharmaceutical industry, in order to avoid any risk of contaminations (both from the chemicals inside the trap and infected mouse blood).

REAL NEEDS

Unfortunately, live-catch traps are very expensive to maintain, because of the labor costs required to continuously check if they need to be emptied. Even though a single mouse catch is a very rare event in today’s hospitals or pharmaceutical depots, all traps must be equally checked every few days - and traps are often located in places that are difficult to access.

This project is designed to cut dramatically labor costs involved by trap checks. In practice you can leave the traps completely unattended until the system calls you for assistance. The system consists of a monitoring station – a computer-controlled receiver with an LCD display and a relay output – and up to 20 mouse sensors. It works placing a mouse sensor – a small plastic box – inside every trap. When a mouse gets captured, the sensor transmits the trap identifier to a master unit, which logs the trap identifier and displays it on the LCD. Here it can be conveniently seen from maintenance people; alternatively the receiver can dispatch the call to an external service, triggering an ordinary automatic phone dialer connected to its relay out.

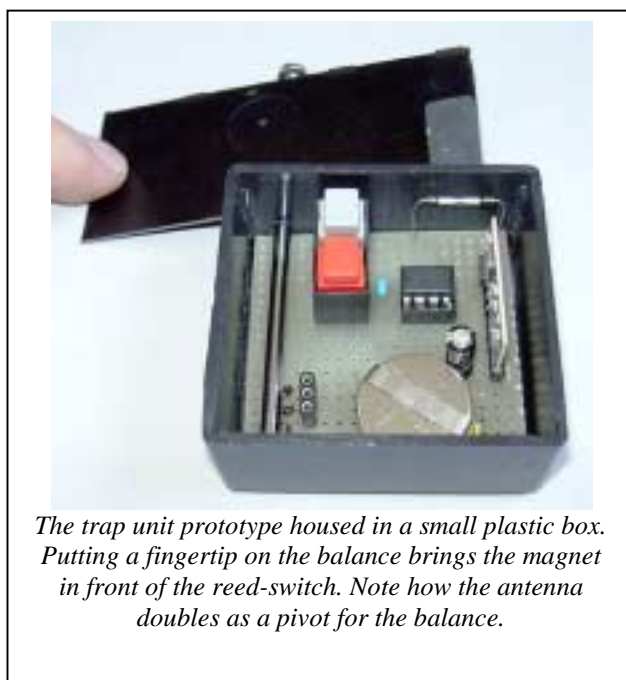
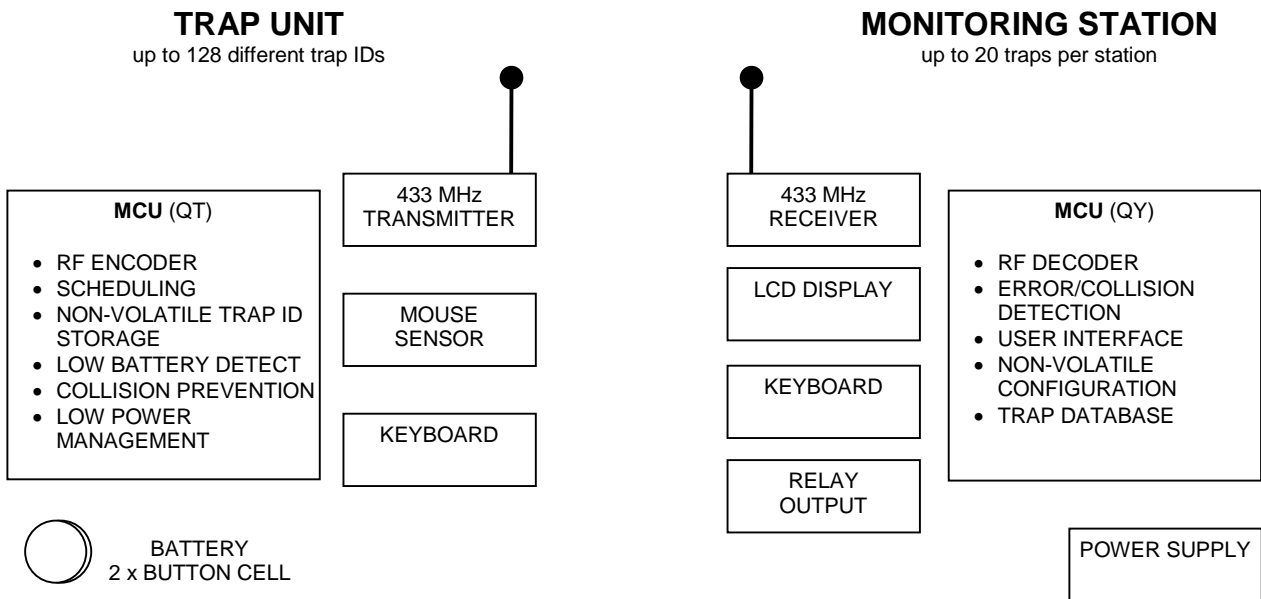


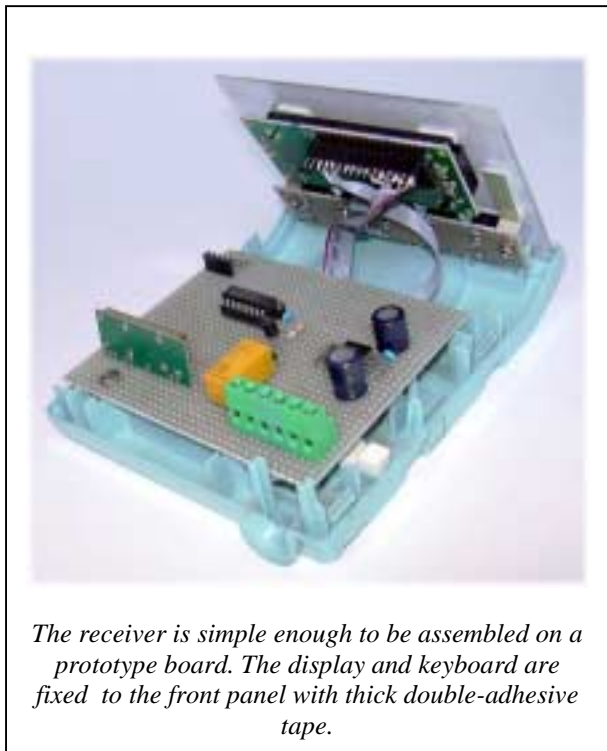
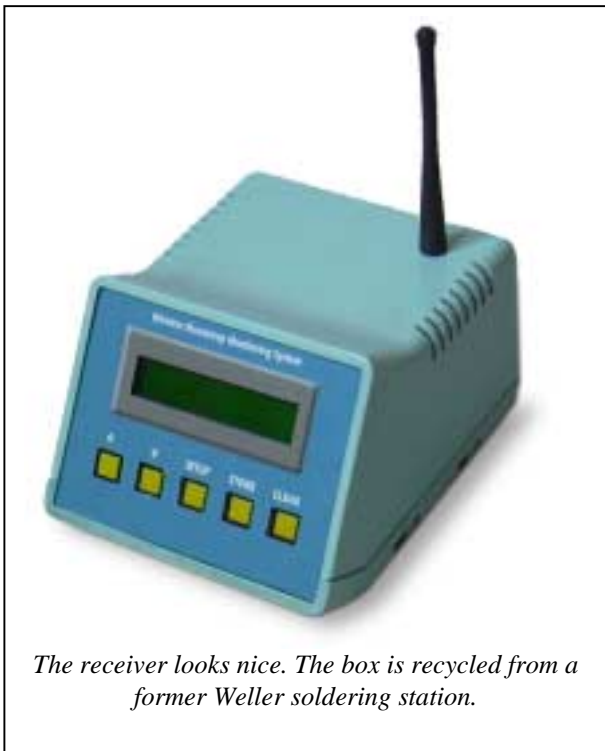
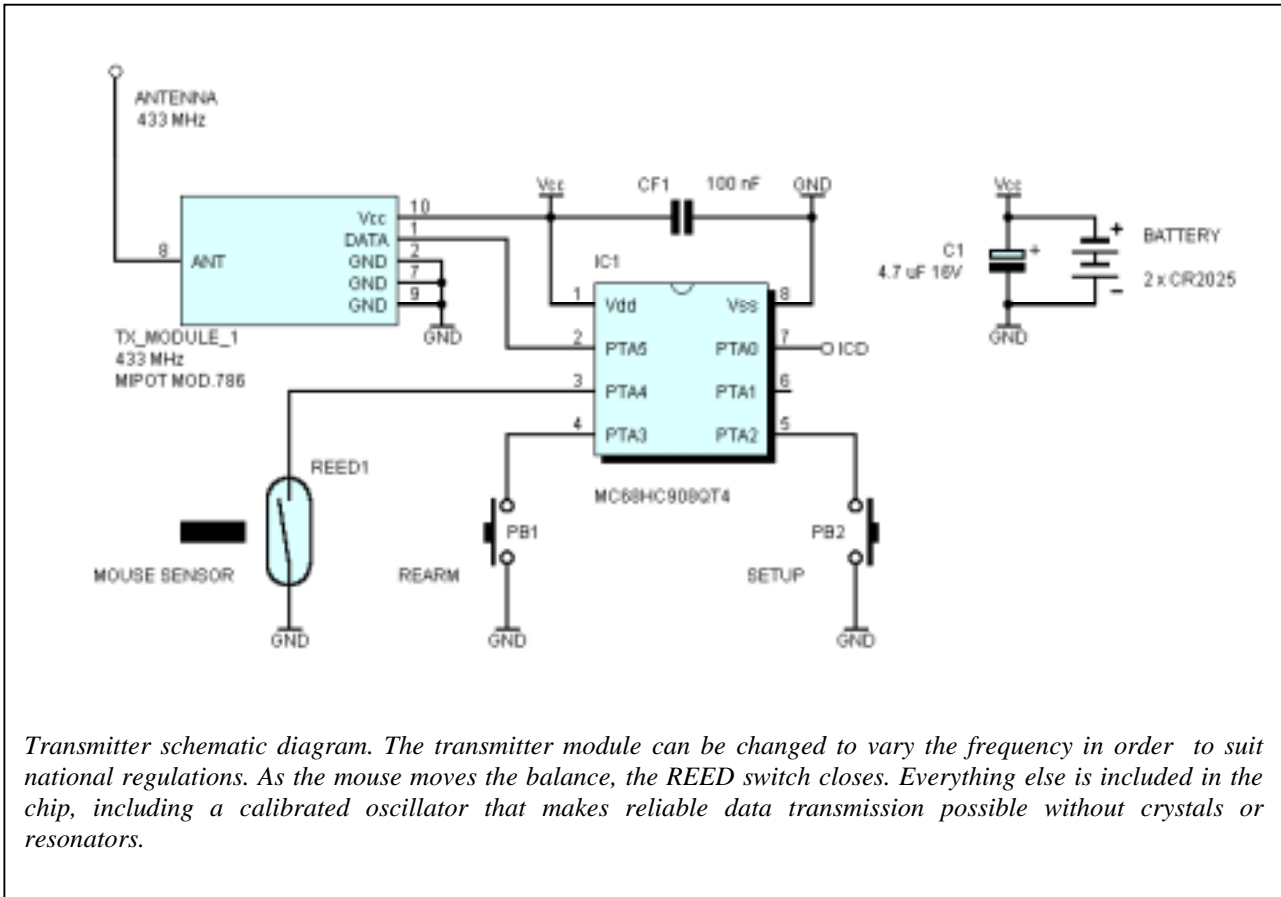
NON-TRIVIAL REQUISITES

Although simple, such a system is a non-trivial design. Parts count must be kept to a minimum, to contain costs as possible. Dimensions are essential as well, to fit even the smallest traps on the market. Mouse sensors need to be battery operated, and batteries must last for years of continuous service. This calls for low-power parts and carefully designed software. The system must resolve collisions due to simultaneous transmission of two or more mouse sensors, must tell the user when

batteries need to be replaced, and must detect when a trap gets lost or destroyed – not an unlikely event in some industrial environments. And, of course, must be simple, resistant to dirt, reliable and easy to manufacture.

On the receiver side, the monitoring station must be cost-effective, dependable and easy to setup and use. It must show complete trap information, and configuration data must be retained after power interruptions. The design must be compact, modular and very flexible: as with all new products, new requirements are expected to grow while the works are in progress, therefore high-level programming languages are preferable.

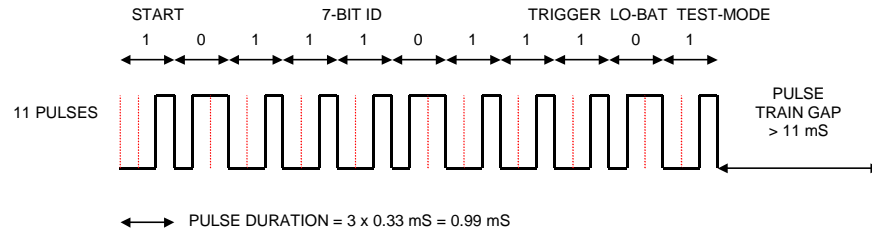




ABOUT SOFTWARE

I wrote the software in C with the free Metrowerks Codewarrior compiler and debugger.

-



```
static void rf_encoder(unsigned int codeword)
{
    unsigned int mask;
    TMOD = CLOCK_FREQ * BIT_THIRD_DURATION; //prepare for timer overflow every .33 us (a bit third)

    mask = 1 << (CODE_BITS - 1);           //prepare mask for filtering leftmost bit
    while( mask != 0 )
    {
        TSC_TSTOP = 0;                       //restart timer
        WAIT_MODE;                             //go in low power mode until a .33 uS period expires
        if ( (codeword & mask) == 0 )         //are we transmitting a zero?
            TX_ON;                             //yes, force a 66% duty cycle
        WAIT_MODE;                             //go in low power mode until a .33 uS period expires
        TX_ON;                                 //set output to ensure a duty cycle be not less than 33%
        WAIT_MODE;                             //go in low power mode until a .33 uS period expires
        TX_OFF;                                //turn transmitter off
        mask >>= 1;                            //prepare mask for filtering next bit
    };
    TSC_TSTOP = 1;                            //stop timer before exit to reduce power consumption
}

```

The signal generated by the `rf_encoder()` function using the 16-bit timer and WAIT mode.

`WAIT_MODE`, `TX_ON` and `TX_OFF` are macros defined in "hardware.h"