

## PageAlert – A Remote Sensing System

Project Number FZ1614

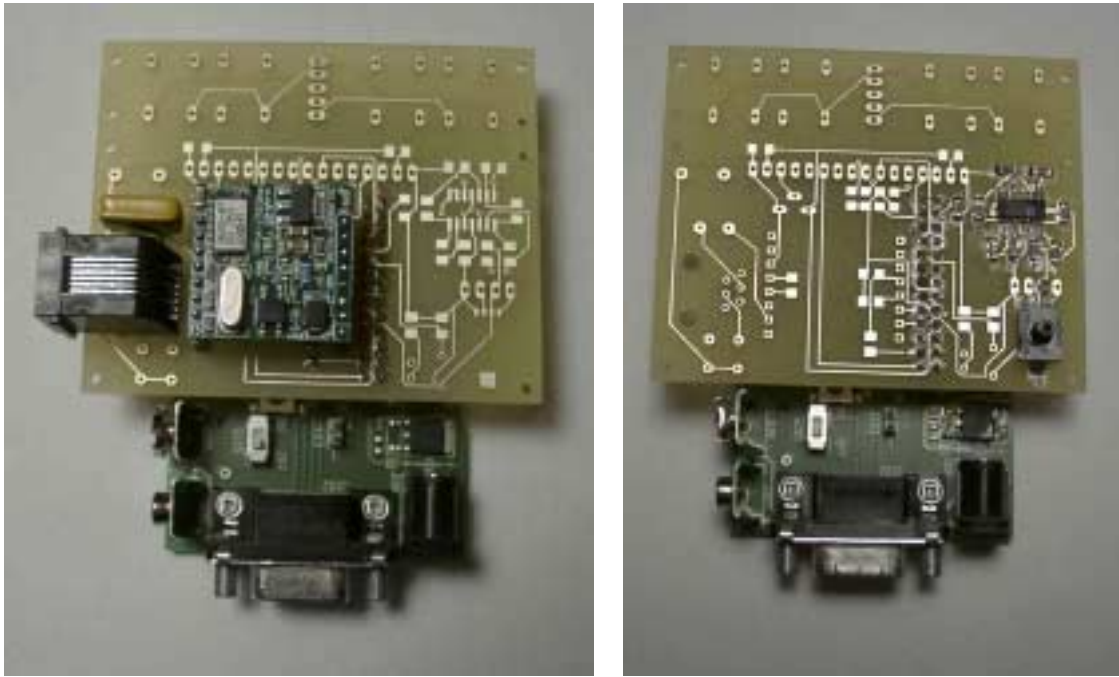


Figure 1 – Base (left) and Remote (right) System

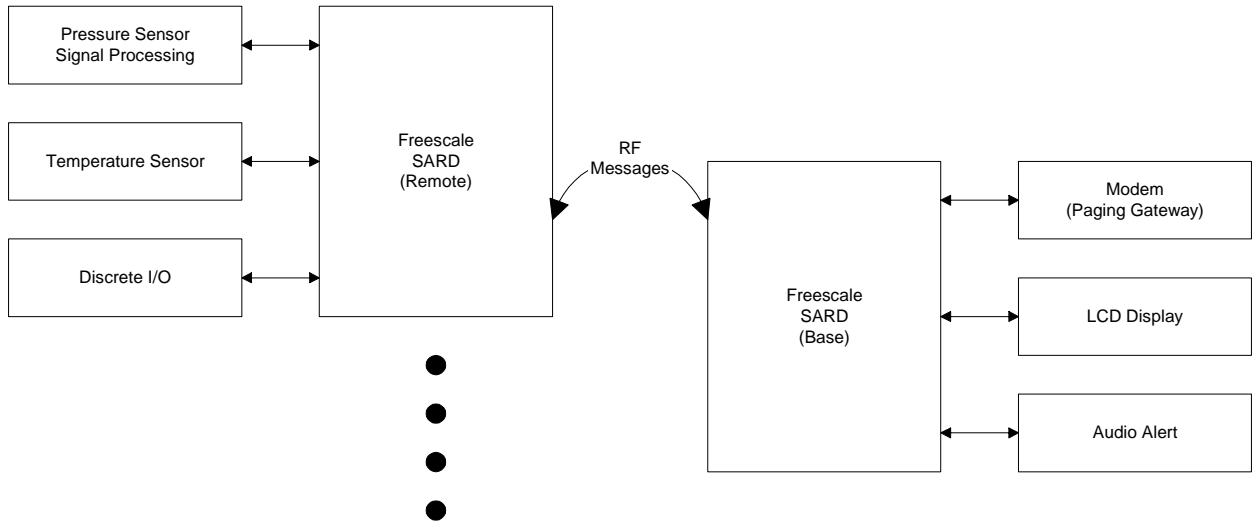
### Description

PageAlert is a flexible remote sensing platform connected to a central data recording and / or alarming system. The Freescale ZigBee platform is ideal for this – A relatively short transmit range, but enough to cover a typical house eliminates sensor wires. General processing and I/O supports an alarm capability.

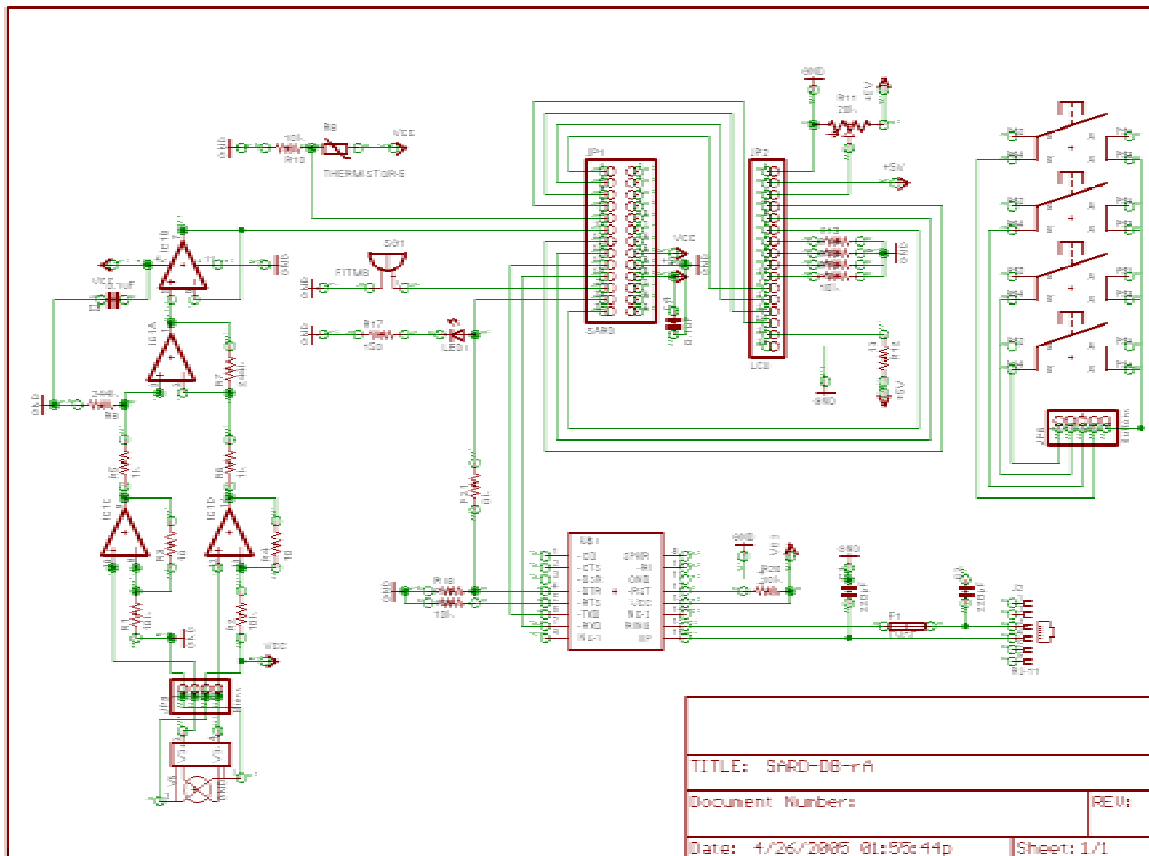
The MPXM2010 pressure sensor can easily sense water levels by connecting it to a dip tube anchored to the bottom of the sump pump crock. The rising water increases the pressure in the tube with full range of the sensor corresponding to about 1 meter of water depth.

The alarm capability is handled by using a 2400 baud modem to connect to the TAP protocol paging service of my pager supplier. By sending a sequence of commands (in manual mode) a page is generated to warn of rising water. A buzzer / audible alarm capability is also desirable.

## Block Diagram



## Schematic



## Code Sample

```
void runQF(void)
{
    QF_init();
    QF_psInit(subscrSto, Q_DIM(subscrSto)); /* init publish-subscribe */

    /* initialize event pools... */
    QF_poolInit((QEvent *)statePoolSto, sizeof(statePoolSto),
                sizeof(statePoolSto[0]));
    QF_poolInit((QEvent *)inputPoolSto, sizeof(inputPoolSto),
                sizeof(inputPoolSto[0]));

    /* start active object(s) */
    MonitorCtor (&monitor);
    QActive_start ((QActive *)&monitor, (uint8_t)1,
                  MonitorQueueSto, MAX_PUB_SIG,
                  (void *)0, 0, (QEvent *)0);
    ChatCtor (&chat);
    QActive_start ((QActive *)&chat, (uint8_t)2,
                  ChatQueueSto, MAX_PUB_SIG,
                  (void *)0, 0, (QEvent *)0);

    MonitorSetChat (&monitor, &chat);

    /* Start up the modem */
    ModemUartCtor (&modemPort);

    QF_run(); /* run the QF application */
}

QSTATE Monitor_Wait (Monitor *me, QEvent const *e)
{
    char tmpStr[32];
    uint16_t sensorValue;

    switch (e->sig)
    {
        case Q_ENTRY_SIG:
            QTimeEvt_fireIn (&(me->timer__), (QActive *)me,
                             NoMsgAlertTime * TicksPerSecond);
            return 0;

        case RFMSGRCVD_SIG:
            Q_DIAG ("Got RF Msg")
            QTimeEvt_rearm (&(me->timer__),
                             NoMsgAlertTime * TicksPerSecond);
            // Check message data against limits
            sprintf (tmpStr, "id=%02x", ((StateEvt *)e)->data[1]);
            Q_DIAG (tmpStr)
            sprintf (tmpStr, "data=%02x%02x", ((StateEvt *)e)->data[2],
                    ((StateEvt *)e)->data[3]);
            Q_DIAG (tmpStr)
            switch (((StateEvt *)e)->data[1])
            {
                case PressSensorId:
                    sensorValue = ((StateEvt *)e)->data[2];
                    sensorValue = sensorValue << 8;
                    sensorValue += ((StateEvt *)e)->data[3];
                    if (sensorValue > PressSensorLimit)
                    {
```

```
        Q_TRAN (&Monitor_Alarm);
    }
    break;
}
return 0;

case TIMEOUT_SIG:
    Q_DIAG ("30 sec msg timeout")
    QTimeEvt_fireIn (&(me->timer__), (QActive *)me,
                    NoMsgAlertTime * TicksPerSecond);
    return 0;

    case Q_EXIT_SIG:
        QTimeEvt_disarm(&me->timer__);
        return 0;
}
return (QSTATE)&QHsm_top;
}
```