

## Supplementary Materials

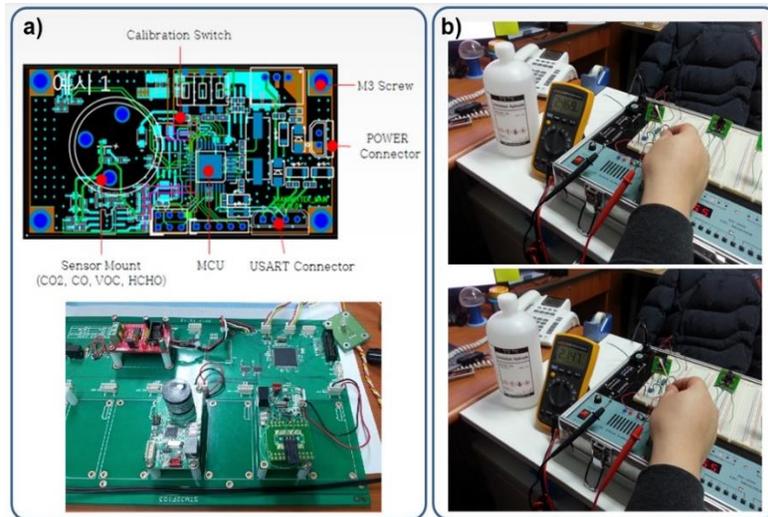


Figure S1. Individual sensor module (a) and lab test for indoor pollutants (b).

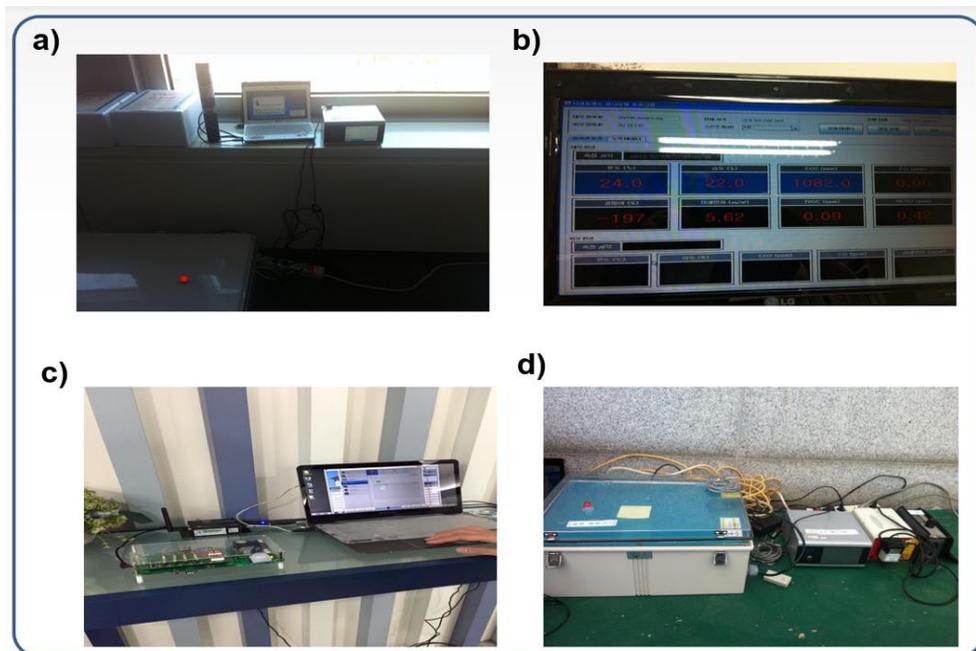
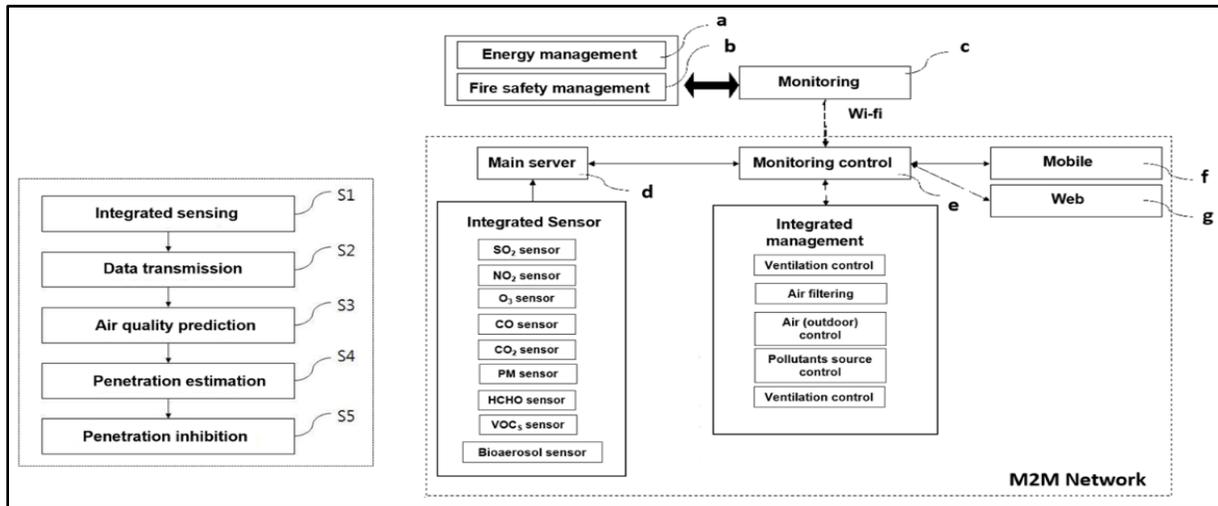


Figure S2. Field test using the integrated sensor platform indoors (a, b, c) and in the underground parking garage (d).



**Figure S3.** Building air quality monitoring (S1–S5) using the integrated sensor platform. Left: block diagram, right: diagram for the indoor air quality management system with an M2M wireless sensor network.

```

a)
void GenerateNetwork(NET* Net)
{
    INT i, j;
    Net->Layer = (LAYER*) calloc(NUM_LAYERS, sizeof(LAYER*));
    for (i=0; i<NUM_LAYERS; i++) {
        Net->Layer[i] = (LAYER*) malloc(sizeof(LAYER));
        Net->Layer[i]->Units = Units[i];
        Net->Layer[i]->Output = (REAL*) calloc(Units[i]+1, sizeof(REAL));
        Net->Layer[i]->Error = (REAL*) calloc(Units[i]+1, sizeof(REAL));
        Net->Layer[i]->Weight = (REAL*) calloc(Units[i]+1, sizeof(REAL));
        Net->Layer[i]->WeightSave = (REAL*) calloc(Units[i]+1, sizeof(REAL));
        Net->Layer[i]->dWeight = (REAL**) calloc(Units[i]+1, sizeof(REAL*));
        Net->Layer[i]->Output[0] = BIAS;
    }
    if (i != 0) {
        for (j=1; j<Units[i]; j++) {
            Net->Layer[i]->Weight[j] = (REAL*) calloc(Units[i-1]+1, sizeof(REAL));
            Net->Layer[i]->WeightSave[j] = (REAL*) calloc(Units[i-1]+1, sizeof(REAL));
            Net->Layer[i]->dWeight[j] = (REAL*) calloc(Units[i-1]+1, sizeof(REAL));
        }
    }
    Net->InputLayer = Net->Layer[0];
    Net->OutputLayer = Net->Layer[NUM_LAYERS - 1];
    Net->Alpha = 0.9;
    Net->Eta = 0.25;
    Net->Gain = 1;
}

b)
void ComputeOutputError(NET* Net, REAL* Target)
{
    INT i;
    REAL Out, Err;
    Net->Error = 0;
    for (i=1; i<=Net->OutputLayer->Units; i++) {
        Out = Net->OutputLayer->Output[i];
        Err = Target[i-1]-Out;
        Net->OutputLayer->Error[i] = Net->Gain * Out * (1-Out) * Err;
        Net->Error += 0.5 * sqr(Err);
    }
}

void BackpropagateLayer(NET* Net, LAYER* Upper, LAYER* Lower)
{
    INT i, j;
    REAL Out, Err;
    for (i=1; i<=Lower->Units; i++) {
        Out = Lower->Output[i];
        Err = 0;
        for (j=1; j<=Upper->Units; j++) {
            Err += Upper->Weight[j][i] * Upper->Error[j];
            Lower->Error[i] = Net->Gain * Out * (1-Out) * Err;
        }
    }
}

c)
void SimulateNet(NET* Net, REAL* Input, REAL* Output, REAL* Target, BOOL Training)
{
    SetInput(Net, Input);
    PropagateNet(Net);
    GetOutput(Net, Output);
    ComputeOutputError(Net, Target);
    if (Training) {
        BackpropagateNet(Net);
        AdjustWeights(Net);
    }
}

void TrainNet(NET* Net, INT Epochs)
{
    INT Year, n;
    REAL Output[M];
    for (n=0; n<Epochs*TRAIN_YEARS; n++) {
        Year = RandomQualINT(TRAIN_LIMB, TRAIN_UPB);
        SimulateNet(Net, &(Sunspots[Year-N]), Output, &(Sunspots[Year]), TRUE);
    }
}

void TestNet(NET* Net)
{
    INT Year;
    REAL Output[M];
}

d)
C:\Wdev\tempWC+WbptestWDebugWbgn.exe
NO    PM10    Open-Loop Prediction    Closed-Loop Prediction
261    0.020    0.046    0.046
262    0.025    0.045    0.048
263    0.037    0.046    0.050
264    0.075    0.048    0.052
265    0.058    0.049    0.048
266    0.082    0.044    0.044
267    0.061    0.053    0.049
268    0.091    0.049    0.046
269    0.081    0.052    0.045
270    0.087    0.052    0.047
271    0.068    0.061    0.051
272    0.071    0.065    0.056
273    0.071    0.067    0.057
274    0.082    0.072    0.063
275    0.124    0.072    0.062
276    0.123    0.079    0.060
277    0.088    0.080    0.058
278    0.061    0.073    0.055
279    0.067    0.059    0.050
280    0.081    0.060    0.051
Press any key to continue.

```

**Figure S4.** Developed prediction program using an Artificial Neural Network (ANN). Initial screen (a), reverse propagation (b), prediction screen (c), and final result (d).