

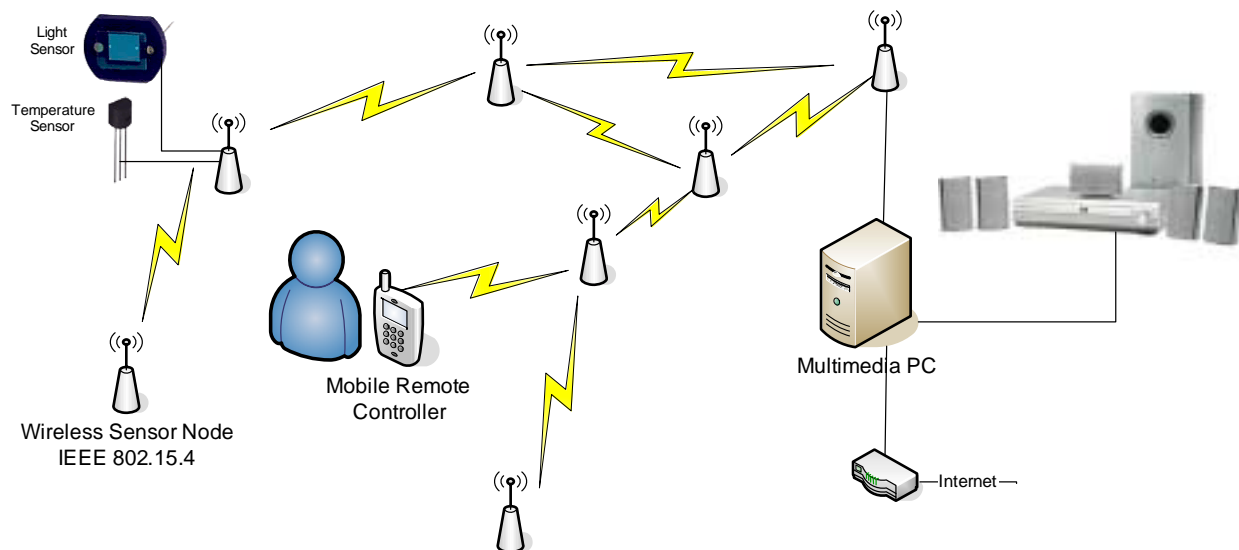
# HAXOR HOME AUTOMATION X OBSERVATION REMOTE

## Abstract

The HAXOR Home Automation and Observation Remote is a general purpose remote control infrastructure designed to integrate into advanced smart spaces such as next generation homes, offices, public venues, and other environments. The system is comprised of a number of wirelessly networked remote control, sensor, and actuator nodes, as well as multi-purpose base stations, and other backbone fabrics that seamlessly work together to provide a number of services to the end-users (See Figure 1).

We present an introduction to and the motivation behind our work and the broad application domains in our strategic aim. We then briefly provide an overview discussion of the system and network architecture, the enabling technologies, and the technological challenges and barriers ahead. In order to solidify our project and provide a tangible prototype, we present the details of an entertainment system remote controller node that relies on a multi-hop IEEE 802.15.4/Zigbee mesh network and serves as the wireless user interface to a PC running mp3 audio playback software. The accelerometers and buttons on the remote control are used as the sensors while an LCD display provides feedback to the user. The underlying mesh network also serves as a light and temperature sensing network that when combined with a centralized controller and the appropriate actuators (e.g. light sources), can manually or autonomously adjust environmental conditions according to user specified demands.

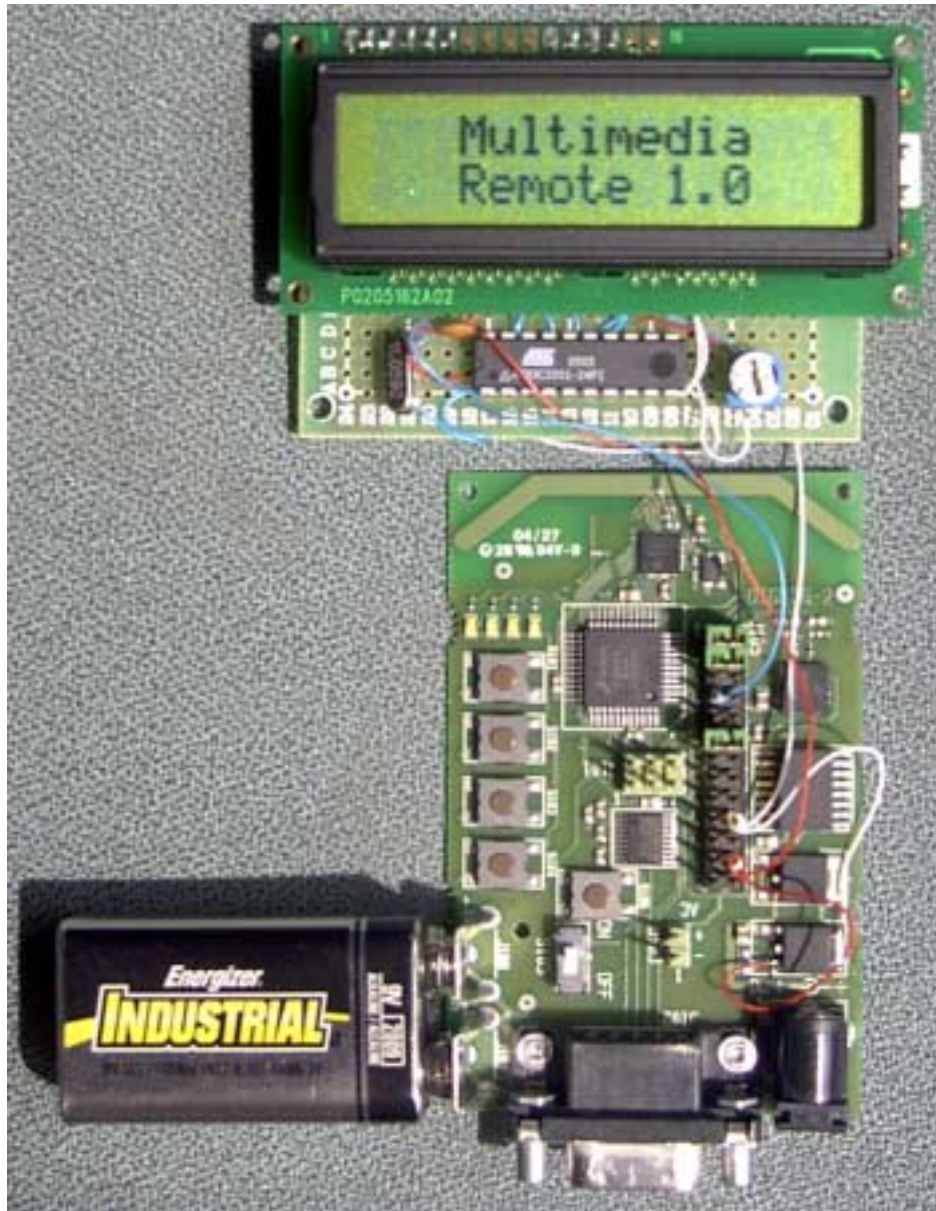
At the center of this prototype is the “remote controller” which is a node specifically designated for this task and outfitted with an LCD screen for user I/O. The commands from the remote controller can be used to control any device or actuator connected to the 802.15.4 mesh network. We demonstrate its use by controlling WinAmp which is an mp3 playback software running on a multimedia PC. We provide functionality such as song selection, play, pause, advance, rewind, and volume control using the various buttons and accelerometer sensors on the node. Side-to-side titling of the remote for example is used to navigate through the song list while front-to-back tilts are used for volume changes. Real time play back information such as song titles and play times are fed back to the remote and displayed on the LCD screen. We must note here that the underlying communication in this network is multi-hop packet



**Figure 1. Overview of the HAXOR infrastructure with IEEE 802.15.4 wireless sensor nodes forming a mesh network. The nodes are equipped with various sensors including accelerometers, light, and temperature sensors. The mobile remote controller is used to control a multimedia PC.**

switching and thus the remote control device can be anywhere in the network as long as a multi-hop communication path (shortest-path) can be found between it and the controlled PC.

In addition to the multimedia control features provided by the controller, the remaining nodes in the network are equipped with light and temperature sensors and provide periodic readings from the sensors. In this case, the data is collected by the PC and logged. This has served as a valuable research tool in building a physical testbed for LC research which we briefly discuss in the project description. We have implemented an actuator box that can be controlled by a PC and set 3 light bulbs at 8 varying intensities each (7 ON states of increasing brightness and 1 OFF). Thus, the remote controller in its current form can trivially be extended to act as a lighting controller by utilizing the PC as a centralized controller. However the details of this step are not part of our contest submission. We should note here that the general problem of distributed modeling of phenomena such as light and their control is an active area of research and provides a number of very interesting and difficult challenges.



**Figure 2. Multimedia Remote Control Module**



Figure 3. Multimedia Remote Control Module LCD screen shots.

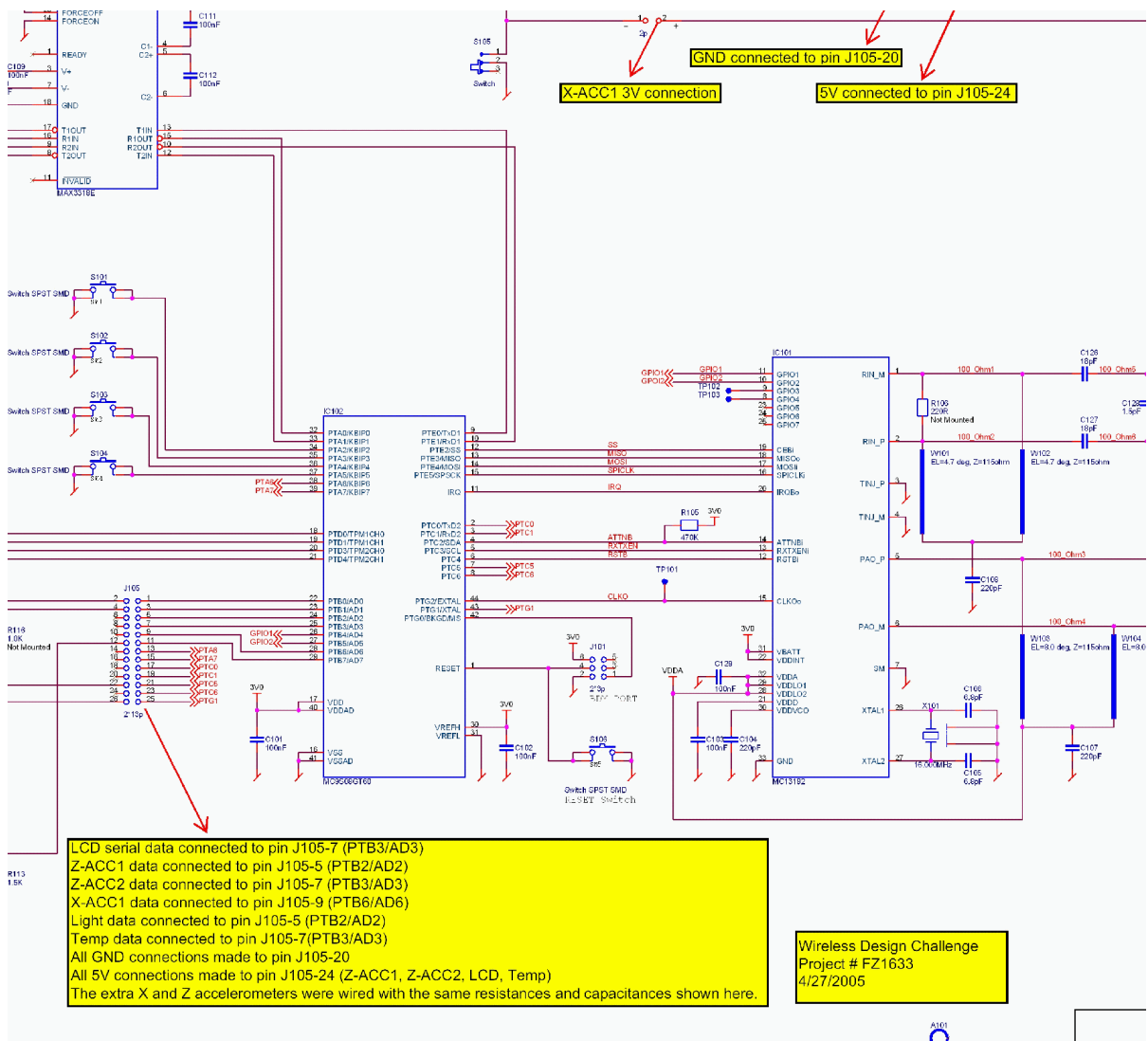


Figure 4. Schematic of the SARD board. For the full schematic, please check the schematic file submitted in the contest entry package.

## Code Example: adjust\_volume() function in the remote controller node

```
void adjust_volume() {
    //find current volume
    request_volume();

    //get initial tilt
    acc = read_xaccel();

    while(PB0 == PRESSED) {

        //go into low power wait
        //periodically resend volume request
        //until we recieve volume or PB0 is
        //no longer depressed
        while(PB0 == PRESSED && volume_valid == FALSE) {
            //set up timer
            TPM2MODH=0x80;
            TPM2MODL=0x00;
            TPM2SC=0x4e;

            //go into low power wait
            _asm wait;

            //we've been woken up, retransmit if volume hasn't been
            //recieved
            if(volume_valid == FALSE) request_volume();
            else {
                //clear the tpm2of flag and disable the timer interupt
                TPM2SC = TPM2SC & 0x3f;
            }
        }

        //get the latest acceleration
        acc_diff = read_xaccel();

        //calculate the difference
        acc_diff = acc - acc_diff;

        //check if we are turning up or down
        //and the tilt is beyond some threshold
        if(acc_diff > 15) {
            //normalize sensor reading
            acc_diff = acc_diff - 16;

            //calculate new volume
            if((volume + acc_diff) > 255) volume = 255;
            else volume = volume + acc_diff;

            //transmit the volume
            tx_data[0] = SET_VOL;
            tx_data[1] = (char)volume;

            tx_packet.dataLength = 2;
            MLME_RX_disable_request();
            MCPS_data_request(&tx_packet);
        }
    }
}
```

```

        MLME_RX_enable_request(&rx_packet, 0);

    } else if(acc_diff < -15) {
//normalize sensor reading
        acc_diff = acc_diff + 16;

//calculate new volume
        if((volume + acc_diff) < 1) volume = 1;
        else volume = volume + acc_diff;

//transmit the volume
        tx_data[0] = SET_VOL;
        tx_data[1] = (char)volume;
        tx_packet.dataLength = 2;

        MLME_RX_disable_request();
        MCPS_data_request(&tx_packet);
        MLME_RX_enable_request(&rx_packet, 0);

    }
    request_volume();
}
}

```