2013 Fifth International Conference on Computational Intelligence, Communication Systems and Networks

Low cost Arduino/Android-based Energy-Efficient Home Automation System with Smart Task Scheduling

Kim Baraka, Marc Ghobril, Sami Malek, Rouwaida Kanj, Ayman Kayssi
Department of Electrical and Computer Engineering
American University of Beirut
Riad El Solh, Beirut, Lebanon
kab06@aub.edu.lb, mmg17@aub.edu.lb, sam41@aub.edu.lb, rk105@aub.edu.lb, ayman@aub.edu.lb

***Abstract*— In this paper, we make use of Home Automation techniques to design and implement a remotely controlled, energy-efficient and highly scalable Smart Home with basic features that safeguard the residents' comfort and security. Our system consists of a house network (sensors and appliance actuators to respectively get information from and control the house environment). As a central controller, we used an Arduino microcontroller that communicates with an Android application, our user interface. Our house network brings together both wireless Zigbee and wired X10 technologies, thus making it a cost-efficient hybrid system. Events can be programmed to be triggered under specific conditions, and this can have a great role in reducing the total energy consumed by some appliances. On the other hand, the system can suggest smart task scheduling. The scheduling algorithm we present is a heuristic for the Resource-constrained-scheduling problem (RCPSP) with hybrid objective function merging both resource-leveling and weighted completion time considerations.**

Keywords- Home automation; Arduino; Android; Smart scheduling; Energy management.

I. INTRODUCTION

In a world which is becoming increasingly automated, Home Automation is acquiring more attention from the households around the world. While many industrial facilities have moved to almost fully automated systems, there are only a few houses that are smartly automated, because of the high cost of such systems. Home Automation should be more accessible since it does not require an incredibly advanced technology and can usually be implemented with off-the-shelf devices. As devices get smarter, a lot of effort is aimed at automating many of our day-to-day activities in a relatively easy way. A new Home Automation field is emerging with the incorporation of the mobile communications technologies into the automation systems, to control appliances either via the existing electrical wiring of the house, or using a wireless network. Indeed, users are using mobile applications on their phones to control their houses from distance. Trivial examples are: turning on the air conditioning system 30 minutes before the resident's arrival, or opening the door lock using a mobile phone. In addition, exploiting such technologies may contribute to a greener planet; and if by a "Smart" Home we mean one that is energy-efficient and that will be able to appropriately manage its energy resources, then this feature is essential for a safer and healthier future.

The objective of our project is to design and implement a remotely controlled and energy-efficient Smart Home. We will be using in our system a low-cost Android tablet as the user interface, and an Arduino microcontroller to offer connectivity with the electronic devices of the house network. The tablet device is the component on which we will develop a user-friendly application, the portal to the house network (either remotely or locally through direct wireless connection to local base). Our house network makes use of both wireless Zigbee and wired X10 technologies, making it a hybrid and flexible system. Zigbee ensures the reliability of the system and X10 allows the integration of low cost controllers to optimize the overall cost of the system (X10 actuators being the cheapest, although not the most reliable). By the appropriate use of sensors, the user can monitor his/her house in real-time (displays for energy consumption, water level, indoor temperature...). This environmental information is also used by the system to trigger events that the user customizes for smart energy management and reduction in wasted energy. Furthermore, some houses running on standalone generators or renewable energy sources may be subject to constraining limits for instantaneous power. This is where automatic task scheduling can play a role in meeting instantaneous power targets, but also cost targets (if prices of energy vary at different times of the day). We will present a heuristic algorithm to the scheduling problem under hand which is an instance of the general resource-constrained project-scheduling problem (RCPSP).

The rest of the paper is organised as follows. Section II presents current works related to our system as a whole, as well as each of its individual components. In section III, we present our design. Finally, Section IV discusses some implementation details.

II. RELATED WORK

A. General Home Automation Products

Numerous Home Automation products exist on the market. Most of them are functional only for sub-products within the brand that sells the system as a whole, but some are compatible with other existing "standard" technologies like X10, KNX, Insteon, Z-wave, etc. The main difference between these products is the interface between user and central controller. Some systems require a software running on PC, other are Web-compatible, making it easy to control your home from any mobile or non-mobile device with Internet connectivity. To cite a few of the systems on the market, Belkin's Wemo allows the control of almost anything in the home through an App. It also can make use of sensors to activate custom events in the house, or use the Web to

978-0-7695-5042-8/13 $26.00 © 2013 IEEE
DOI 10.1109/CICSYN.2013.47

notify the user about events happening in the house using the IFTTT (If This Then That) service [1]. The magDomus system is another solution for Home Automation; this one is compatible with most existing technologies, which is very practical. The most popular computer software for general Home Automation applications (an open-source software) is MisterHouse, which is customizable, very flexible and compatible with most technologies. [2]. However, it runs only on Windows and Linux and is not compatible with Arduino.

B. Central controller (or brain of the system)

Some Home Automation projects have been designed for sending commands within the home only. For this purpose, commands may be received from a short range wireless technology like Bluetooth. This is the case in [3], where the commands are sent from a Smartphone through Bluetooth to an Arduino Mega board as a central controller. Another option is to connect the microcontroller to the Internet to be able to communicate with it from any device that has Internet-connectivity. This can be achieved by the presence of a Web server. The Web Server can run directly on the microcontroller, or on a separate PC with connectivity to the microcontroller, as depicted in [4] or on a router running OpenWrt. One other option is to use cloud services like Cosm (previously called Pachube), whose services (storage of sensor data on cloud servers) can relieve the microcontroller's memory and computational power.

C. Communication protocols between Home Automation devices (or brawn of the system)

When designing a Smart Home Automation environment, one of the essential concerns is to decide on the communication protocol (or standard) that will link the "brain" to the different devices of the system. We distinguish three kinds of protocols.

Direct connection

The most basic approach is to use a direct connection of the devices to the microcontroller, like in [4]. Such a procedure is very undesirable in our case, since the Arduino board represents our central unit and not a gateway between central unit and devices like in this project. This would offer very low scalability (limited to the number of output pins of the Arduino) and would need extra infrastructure or rewiring.

Wired protocols

X10 is one of the most popular protocols in the Home Automation business, with millions of homes using it around the world, mainly due to the cheap accessibility of its components. Furthermore, most of its units are extremely easy to set up ("plug and play"). In [5], X10 devices are controlled from an Arduino board, making use of the built-in X10 Arduino libraries to send X10 messages through output pins.

The X10 protocol strength lies in its easiness of use and compatibility with Arduino. However, in [6] the restrictions of X10 are presented. First, collisions may appear if one is using several controllers or repeaters, and X10 doesn't possess the capacity to detect such collisions. This may sometimes lead to altering the status of several devices for no reason. Second, the protocol is considered to be relatively slow. Third, the X10 signal may experience attenuation when there is electrical noise, which is not infrequent. As a result, one of the devices can be accidentally switched on. As a conclusion, X10 should be used for applications that are error-tolerant like lighting for instance, but not for applications like security.

Wireless protocols

Since most RF devices that work on a remote control operate at a frequency of 433 MHz (European Standard), some DIYers have considered the option to reverse-engineer the ASK modulation scheme used to create Arduino libraries that emulate the functioning of the remote control. A system with this capability can mostly control remote sockets, but it is not very flexible and its scalability is questionable.

Other wireless options that are "cleaner" (no hacks involved) include the wireless standard that has succeeded to the old X10 wired technology, which is called Z-Wave. Since it is relatively new, no low cost projects have been found using this technology and it is nowadays mostly integrated in market solutions or products development. Z-Wave is not a completely license-free protocol and may be quite expensive compared to other wireless options.

Another option for a wireless communication within the house network is the well-known Zigbee protocol, like in [6]. Their system uses a mesh-network topology for greater robustness, by making use of the capability of the nodes to self-organize themselves into a mesh network. As a wireless protocol, Zigbee provides one of the most efficient ways to integrate a large number of wireless nodes in a Home Automation setting.

D. Energy Management

People nowadays are aware of the necessity of managing their energy consumption in a more reasonable way but efforts to succeed in doing so remain not as easy as we think they are. Home Automation can solve this issue and make a significant impact concerning an improved energy organization [7] [8]. There exist several aspects of our houses that we can improve using Home Automation technologies [8]. If we consider the house's lighting, for example, dimming the lights or making their intensity adjustable according to the activity carried out, can yield a considerable reduction in the amount of electricity payment. The same applies for the heating and cooling systems. Home Automation can also help monitoring the water consumption, especially when it comes irrigation.

E. Scheduling Algorithm

The general RCPSP can be formulated as follows: given n tasks $\{T_1,...,T_n\}$ sharing a common resource with resource utilization $\{U_1,...,U_n\}$, durations $\{D_1,...D_n\}$ and weights $\{W_1,...,W_n\}$, we want to find a schedule $\{S_1,...,S_n\}$ (where S_i is the starting time of task T_i), such that the objective function is optimized, under the constraint that the total resource usage

U[t] is less than *U* (resource upper limit) at all time slots t. [12]. The objective function to be minimized can vary depending on the application and may be the total weighted completion time, the total weighted tardiness, the variance in resource utilization or other non-regular objective functions [10]. The tasks may also have release dates hence the need for an online algorithm. Pre-emption (stopping a task while it is executing) as well as precedence between tasks may or may not be allowed. In our case, we will assume that all tasks are known ahead of time and that no precedence relation exists between tasks. We will be looking for a non-preemptive offline solution to minimize both the variance in resource usage and the weighted completion time.

It has been proven in [11] that the RCPSP problem is NP-hard. Optimal solutions to this problem have been found using dynamic programming [13] and branch-and-bound [14] approaches. Some heuristics have been devised for the RCPCSP with weighted completion time as the objective function. These include: priority-rule based methods, truncated branch-and-bound or integer programming methods [15].

III. Design

A. Central Controller

A central controller (our Arduino board) receives user commands to execute. It has Internet connectivity through an Ethernet shield mounted on the Arduino. On the user side, a mobile device provides interface with the system as a whole through a user-friendly application. The mobile device can be either wired to the central controller (through USB cable for instance), or communicates with it wirelessly. Within the scope of the home, wireless connectivity can be achieved using an Ethernet shield on the central controller. This way, we would be able to access the controller either locally or remotely through the Internet.

In our case, the client-server architecture is the one to opt for, since the central controller acts as a fixed entity that responds to clients' (mobile devices) requests (and eventually sends them notifications as well). Hence the need for a server (at the application level, i.e. a piece of code that is able to respond to client requests) closely tied to the central controller. We will use a simple Web server application running on the Arduino that communicates through the HTTP protocol with our Web-based Android application.

B. End-nodes

End-nodes within the house network can be categorized into two separate types: they are either sensors that send environmental information (temperature, luminosity, motion, humidity, ...) or actuators that perform specific tasks (switching, dimming, ...). These nodes constitute a network that can be wired, wireless, or hybrid. Topologies can vary and have to be wisely chosen to optimize traffic load and reliability. Figure 1 below shows all the necessary components in our system. Each project component or group of components can be designed and implemented in many different ways. Figure 2 below shows a functional

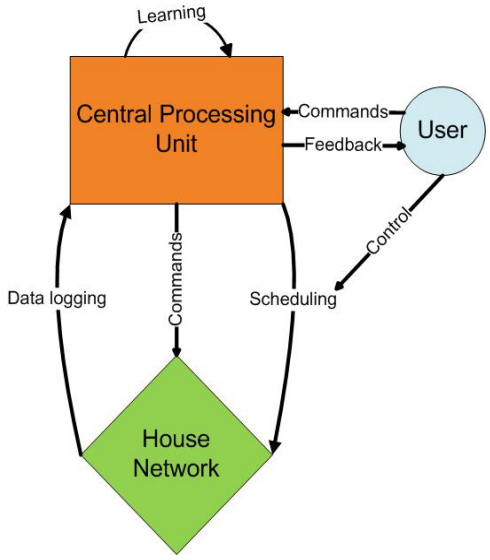

Figure 1: Functional diagram of the system.

block diagram of our system. It embodies the main operations that will be performed by the distributed software on the mobile device/central controller/house network nodes.

For the communication protocol, we used a combination of ZigBee and the simple X10 (specific) protocol which gives our system more flexibility and increases its scalability. Indeed, ZigBee, unlike Z-wave, INSTEON, and other IP-based solutions were designed for general purposes. The Xbee chips running the ZigBee stack will constitute the main nodes in our wireless network.

In our project, communication between two end-nodes is not crucial. In fact, by nature, the home automation framework is highly centralized. We may have conditional actions between two end-nodes for instance (for example switch the light on when someone is approaching), but since the central controller has to have knowledge about the status of each end-node, hopping to the Coordinator node is inevitable. This is why we chose to implement a tree topology, where routing nodes have the role of hopping information to faraway end nodes, and possibly, aggregating sensor data coming from different sensors to send them in a single packet (and hence reduce traffic load in the network).

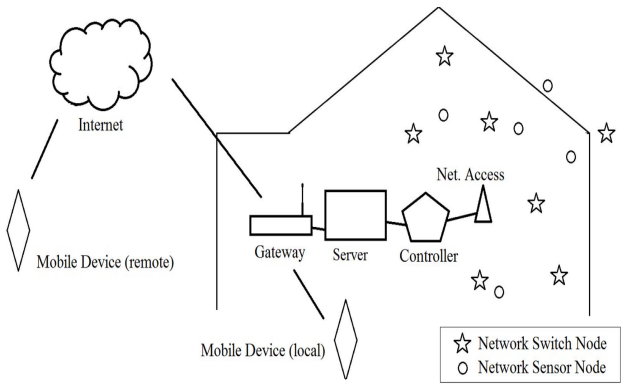

Figure 2: High-level Schematic.

C. Energy Saving

Building on the infrastructure described above, we can use automatically triggered events for energy saving purposes. At any point in time, the Arduino contains a set of conditions and events to be triggered if these conditions are satisfied. These conditions can be either environmental conditions based on sensor readings (for instance turning lights off when luminosity exceeds a threshold) or based on appliance statuses (turn off appliance A whenever appliance B is off).

D. Scheduling algorithm

Now we turn our attention to our scheduling algorithm. Take *{Ti}, {Ui}, {Di}, {Wi}* and *{Si}* as mentioned above. We will refer to Ui as the power rating of task Ti and Wi as its priority value where Wi $\in$ {1,2,3,4,5}. We assume that the electricity pricing is in $/kWh regardless of the instantaneous power usage; we also assume different day and night tariffs.

D.1 Input

The input to our algorithm is summarized as follows:

- $\{T_i\}$: list of tasks,
- $\{U_i\}$: list of power ratings of T_i,
- $\{D_i\}$: list of durations of T_i (in number of time slots needed),
- $\{W_i\}$: list of priorities of T_i,
- *{Forecast[t]}*: Power usage forecast for the day,
- *C(day); C(night)*: Electricity pricing (in $/kWh) during the day and night, respectively and
- *Targ*: Daily cost target (in $/day).

We assume that the algorithm will attempt to schedule the T_i's in the 24 hours following the time the algorithm is run. Time is divided into intervals of length Δt. We take Δt=15 min. *Forecast[t]* is an estimate of the baseline of power usage at time t. It is obtained as follows. The user labels every appliance in the house by either a permanent tag or a variable tag. These can appliances that are always ON like the fridge or the security system, or may vary during the day with pseudo-regular patterns (on the short run) like the air conditioning system or the water heater. Assuming each permanent appliance has a power sensor measuring its consumption, *Forecast[t]* is obtained by adding all power values of permanent appliances at time t, based on history.

D.2 Objective function and constraints

Let *WC* denote the total weighted completion time ($WC = \sum S_i W_i$.) Let σ^2 denote the variance in power consumption throughout one day ($\sigma^2 = Var[Forecast[t] + \sum U_i X_i[t]]$) where $X_i[t]$ is an indicator variable indicating if task T_i is scheduled or not at time slot t. $X_i[t]$ equals1 if $S_i \leq t < S_i + D_i$ and equals 0 otherwise). The objective function (to be minimized) is a weighted sum of *WC* and σ^2: $\alpha.WC + (1-\alpha).\beta.\sigma^2$, where α is a parameter set by the user to give more or less importance to the early finish of higher-priority tasks (the lower α is, the more importance will be given for the maximization of the smoothness of the power usage curve or minimization of the σ^2 term in the objective function. β, on the other hand, is a normalizing parameter to make *WC* and σ^2 compatible, these quantities having different units (β is chosen such that, setting α to 0.5, we obtain a balanced emphasis on both *WC* and σ^2. β can be thought of as the cost of sacrificing one unit of *WC* for one unit of σ^2.). The power and cost constraints are summarized as follows.

1. The overall cost must be under the value of *Targ* inputted by the user.
2. The power usage in any time slot t should never exceed the hard limit U.

D.3 Procedure

Next, we present the suggested heuristic for solving approximately the problem stated above. Although the problem is not greedy in nature (and this is why it is NP-hard), we believe a greedy approach is a reasonable option for an approximate solution to our problem, especially that the objective function is a combination of two terms with possibly opposite gradients (one may increase when the other decreases). The procedure goes as follows:

1. *First schedule tasks with W_i=5 starting with task with highest P_i and going to the lowest P_i value, by identifying the first slot that can accommodate P_i. If two tasks have the same P_i, prefer longer tasks.*
2. *Sort remaining tasks by decreasing order of P_i, and iteratively schedule individual tasks as follows: for each T_i, temporarily assign a start date S_i and compute the resulting objective function. Repeat for all possible S_i values (from 0 to 95, since we considered a 24 hours time span divided into 15 minutes slots), and schedule T_i at S_i^* which results in lowest value of the objective function, and such that the power limit condition is not violated. Terminate with negative output if a point is reached when a given T_i cannot be scheduled.*
3. *If a feasible schedule is found, compute its overall cost C and compare to Targ. If C>Targ, shift tasks with the lowest priority to the night period by rerunning the algorithm in this time period, starting with tasks requiring the highest amount of energy D_iP_i.*

IV. Implementation And Results

We built a prototype of our design described in section III. In this prototype we combined the different building blocks on the physical and application layers. Specifically, we implemented an Android application from the user perspective. We also implemented a Zigbee network on the appliances side controlled by the Arduino through the application interface. We tested the overall design point-to-point communication between two Xbees as well as the overall network connectivity.

A. Android Application

An Android application (Figure 3) was programmed to allow the user to set up a network of appliances actuators and

sensors. The application supports both X10 and ZigBee protocols, in accordance with our design decision. In addition, we did not limit the application functionality for Home Automation, but rather for any kind of Network, thus increasing its scalability. The figure below illustrates a sample snapshot of the device configuration page in our application. The application allows user to add elements to the network, for example sensors in the figure above, and assign addresses as well as choose the type of the network element.

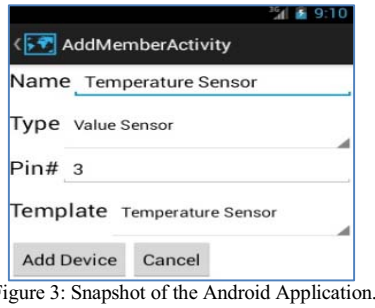

Figure 3: Snapshot of the Android Application.

B. *Arduino programming and electronics*

For our prototype, we used an Arduino MEGA board with Ethernet shield to provide local network connectivity (Android tablet connected on the same network wirelessly). We programmed the Arduino and the Xbees to enable the following scenario:

1. The user sends commands from the Android tablet to the Arduino board.
2. The Arduino, in turn, transmits the instructions to a network of Xbees formed by a router, a coordinator node and an end node.
3. Xbees send sensor readings from the appliances to via the Arduino.
4. The Arduino has the capacity via the presented scheduling algorithm to propose suitable operating schedule based on resources and availability.
5. Scheduling information together with sensor information collected from the Xbees appear on the Android application GUI. Databases for different users will be updated accordingly.

To elaborate, the Arduino contains a database (stored locally on a SD card mounted on the Ethernet shield) with the list of devices connected and their current status. The Android App is synchronized with the content of this database. Some automated events are triggered based on the statuses stored in the database. The Arduino board also runs the scheduling algorithm explained earlier and sends to the Android device for user approval. Once a schedule is obtained for execution, it is stored locally on the Arduino in a table that is periodically checked to trigger events at specific times. We implemented our code using built-in Arduino and Android functions libraries to support the desired interface. This also included a Web Server on the Arduino board that is able to connect with clients and send sensor information as an HTTP command. We also adapted some code from variety of resources.

C. *Scheduling Results*

In the actual system , Δt=15 min and the number of time slots for scheduling is 96; however, for the sake of illustration, in our simulation, these parameters will be respectively set to 30 min and 14. As our power baseline ({Forecast[t]}), we took the following permanent tasks:

- Fridge: Always ON with a power rating of 200 W - Room Air Conditioning: ON from slot 3 to 7 with a power rating of 1,100 W - Laptop: ON from 1 to 10 with a power rating of 100W.

Table 2 shows the schedule obtained after running the algorithm. Note how tasks 1,2 and 3 where rescheduled when to meet the new power constraint U=2,000 W. In Figure 4, the power baseline and the overall power after scheduling are plotted.

TABLE 1: SAMPLE LIST OF TASK TO BE SCHEDULED

Task ID	Task Description	Duration (time slots)	Priority (1-5)	Power rating (W)
1	Well Pump	3	5	800
2	Water Heater	5	4	540
3	Clothes Dryer	2	3	420
4	Dehumidifier	4	2	350
5	Dishwasher	2	4	1200
6	Weed Eater	2	1	500
7	Iron	1	2	1100
8	Vacuum Cleaner	1	3	600

TABLE 2: OBTAINED SCHEDULE FOR TWO VALUES OF POWER LIMIT U

Task	1	2	3	4	5	6	7	8
U=2,500 W								
Starting Slot	0	5	7	7	10	13	12	13
Ending Slot	2	9	8	10	11	14	12	13
U=2,000 W								
Starting Slot	7	0	0	4	10	13	12	13
Ending Slot	9	4	1	7	11	14	12	13

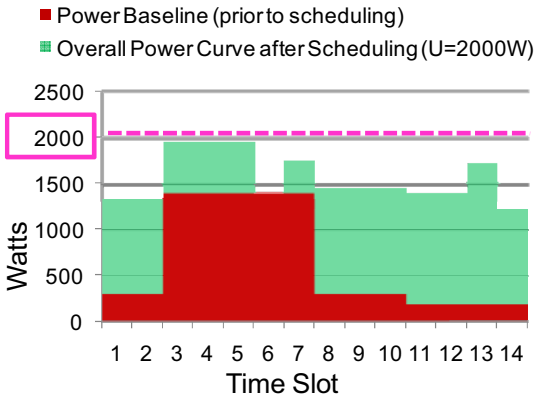

Figure 4: Power curves for the simulated scenario with power limit U=2,000W.

D. Summary

In conclusion, we have proposed a system that combines, on the infrastructure side, both wired (x10) and wireless (Zigbee) technologies for home automation purposes. The system is highly flexible and scalable and can be expanded to large households. On the software side, our Android application ensures that the system enables energy saving, and can suggest task scheduling with both instantaneous power and cost considerations. Figure 5 illustrates the final design picture highlighting the infrastructure and different interfaces.

ACKNOWLEDGMENT

We thank the ECE Dept at the American University of Beirut for sponsoring this work. We would also like to thank Intel Corporation for providing us with the Android tablet.

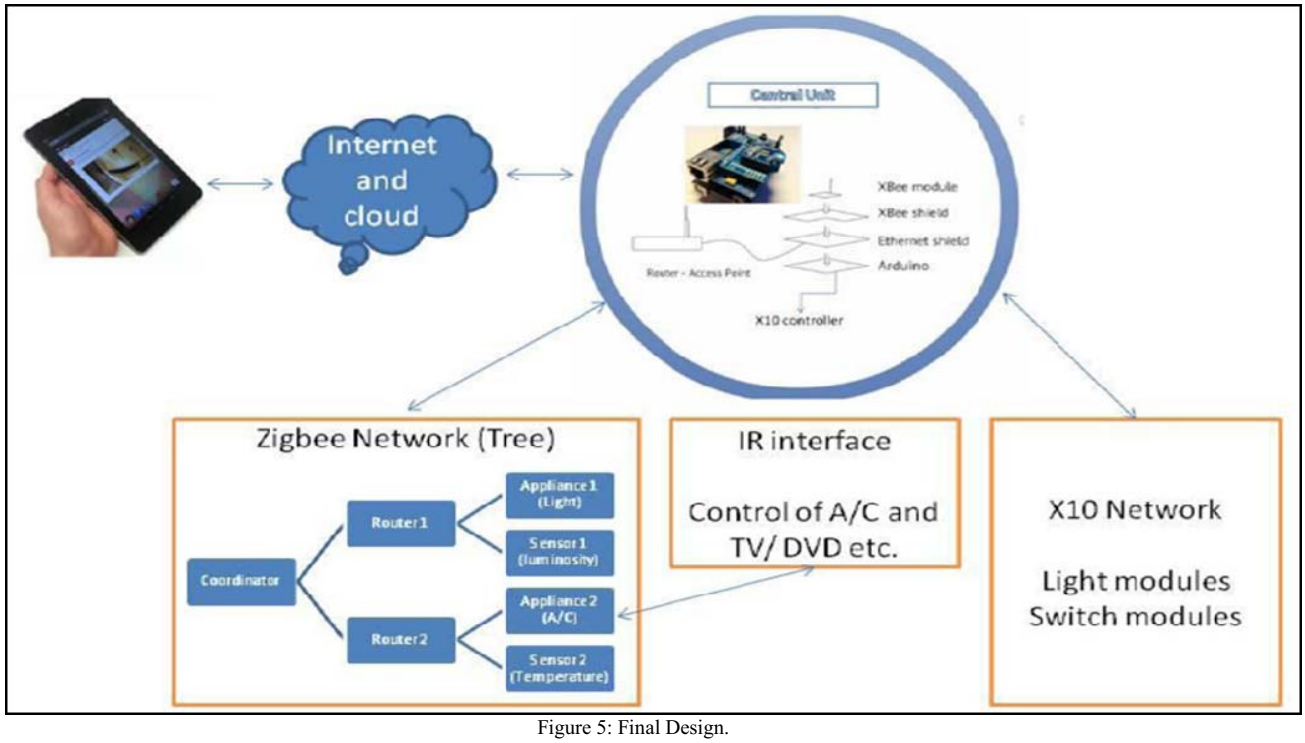

Figure 5: Final Design.

REFERENCES

[1] "What can Wemo Do?," [Online]. Available: http://www.belkin.com/us/wemo.

[2] "MisterHouse -- It Knows Kung-Fu," 3 12 2008. [Online]. Available: http://misterhouse.sourceforge.net/.

[3] I. Petrov and S. Petrov, "uControl Solutions".

[4] A. ElShafee and K. Alaa Hamed, "Design and Implementation of a WiFi Based Home Automation System," World Academy of Science, Engineering and Technology, 2012.

[5] D. Banks, "Home Automation Iphone Application," 2010. [Online]. Available:http://www.utm.edu/departments/engineering/documents/Home Automation using x10 network.pdf. unpublished.

[6] J. A. Infantes Diaz, "Wireless Sensor Networks Controlled With PIC Microcontrollers and Zigbee Protocol," Mikelli University, May 2012.

[7] R. J. C. Nunes, "Home Automation - A Step Towards Better Energy Management," [Online]. Available: http://www.icrepq.com/pdfs/CALEIRA416.PDF

[8] N. Banerjee, "Automating Energy Management in Green Homes," [Online]. Available: http://conferences.sigcomm.org/sigcomm/2011/papers/homenets/p19.pdf.

[9] C. Gomez and J. Paradells, "Wireless Home Automation Networks:A Survey of Architectures and Technologies," IEEE Communications Magazine, 2010.

[10] B. Yang, J. Geunes and W. O'Brien, "Resource-Constrained Project Scheduling: Past Work and New Directions," unpublished.

[11] J. Yuan, T. Cheng and C. Ng, "NP-hardness of the single-variable-resource".

[12] P. Brucker, "Scheduling and constraint propagation," *Discrete Applied Mathematics,* vol. 123, no. 1-3, pp. 227-256, 2002.

[13] I. STANIMIROVIĆ, M. PETKOVIĆ, P. STANIMIROVIĆ and M. ĆIRIĆ, "Heuristic Algorithm for Single Resource Constrained Project scheduling Problem based on the dynamic programming," *Yugoslav Journal of Operations Research,* vol. 19, no. 2, pp. 281-298, 2009.

[14] R. Kolisch and S. Hartmann, "Heuristic Algorithms for Solving the resource-constrained Project Scheduling Problem: classification and computational analysis".

[15] P. Brucker, et al. "Resource-constrained project scheduling: Notation, classification, models, and methods," *European Journal of Operational Research,* vol. 112, no. 1, pp. 3-41, 1999.