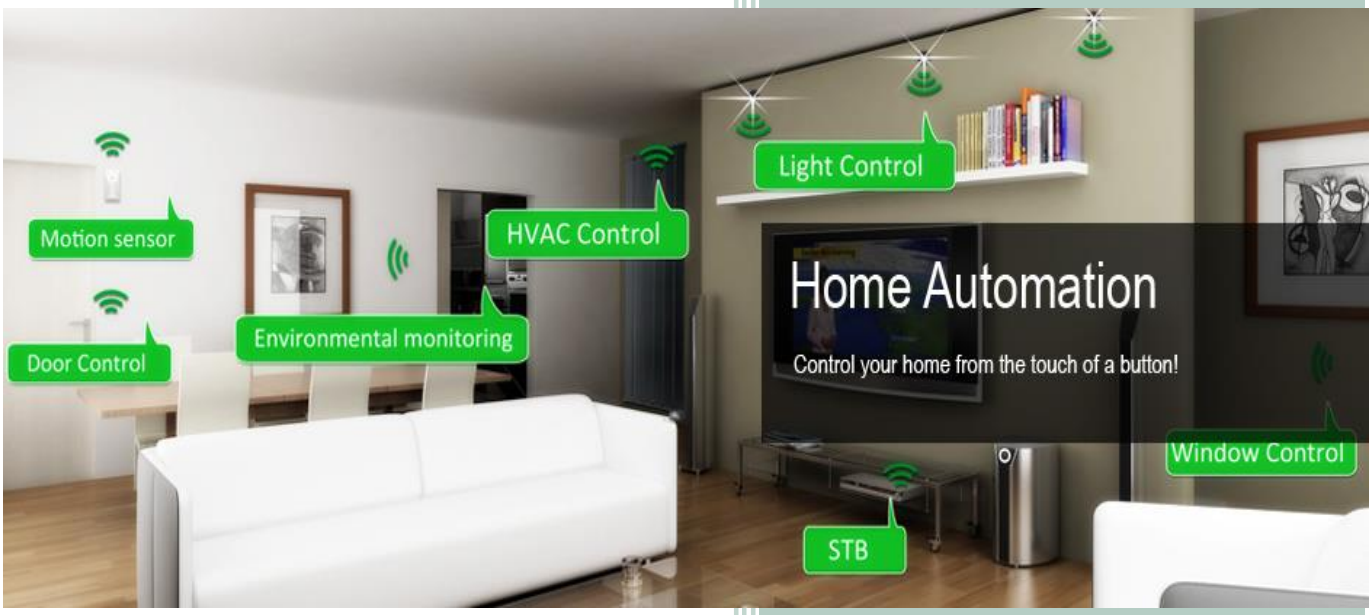
2014

HOME AUTOMATION



Trishit Dutta (ECE)

Mohit Rathore (ECE)

Adrish Chatterjee (ECE)

Divyank Goswami (ECE)

What is 'HOME AUTOMATION'?

Home automation refers to the use of computer and information technology to control home appliances and features (such as windows or lighting). Systems can range from simple remote control of lighting through to complex computer/micro-controller based networks with varying degrees of intelligence and automation. Home automation is adopted for reasons of ease, security and energy efficiency.

As the number of controllable devices in the home rises, interconnection and communication becomes a useful and desirable feature. For example, a furnace can send an alert message when it needs cleaning, or a refrigerator when it needs service. If no one is supposed to be home and the alarm system is set, the home automation system could call the owner, or the neighbours, or an emergency number if an intruder is detected.

In simple installations, automation may be as straightforward as turning on the lights when a person enters the room. In advanced installations, rooms can sense not only the presence of a person inside but know who that person is and perhaps set appropriate lighting, temperature, music levels or television channels, taking into account the day of the week, the time of day, and other factors.

Other automated tasks may include reduced setting of the heating or air conditioning when the house is unoccupied, and restoring the normal setting when an occupant is about to return. More sophisticated systems can maintain an inventory of products, recording their usage through bar codes, or an RFID tag, and prepare a shopping list or even automatically order replacements.

Home automation can also provide a remote interface to home appliances or the automation system itself, to provide control and monitoring on a smartphone or web browser.

The need

Necessity may be the mother of invention but laziness is much more motivational than necessity. That said, Home Automation is not totally unnecessary. We are already using automation in our everyday lives. Starting from the TV remote in the living room to the microwave oven in the kitchen or washing machine in the bathroom. The wider approach is to integrate all of the functionalities together neatly into a well-designed and networked system. Also, it makes it easier for disabled/elderly people to remain in control of their home.

Our Implementation

We implemented home automation as a proof of concept. It has six elements that can be manually controlled and a sensor that gives temperature and humidity data. Four of the elements, namely the CFL, AC, tubes and incandescent bulbs can be turn on or off only. As CFL, A/C and tubes are impractical for demo purposes, they are emulated using incandescent bulbs.

The remaining two are DC fans, and have control over intensity by PWM. Although fans in our homes use AC, but since the circuit is a bit complicated to design and implement, so for demo purposes, we are using DC fans.

The first four are implemented using relays and the fans are controlled using MOSFETs.

Although a more permanent method of control would be an android app or tablet, but for the sake of simplicity a locally hosted webpage is our chosen form of interface.

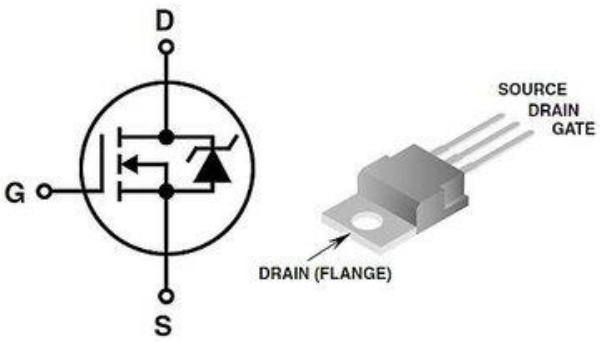
Parts used:

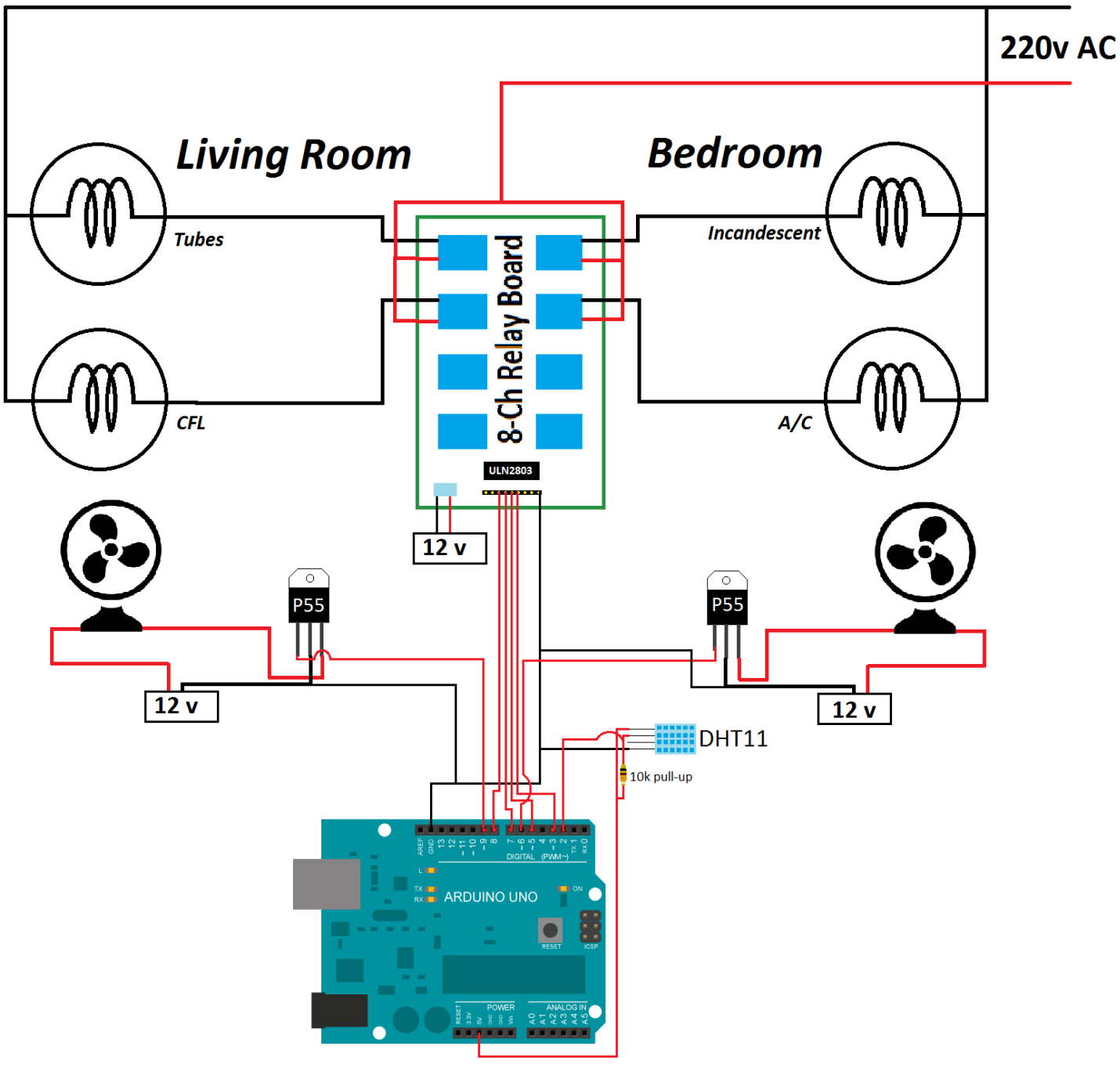
1. Arduino Uno R3 with ATmega 32
2. Ethernet shield with microSD card slot
3. 8-Ch relay board
4. P55 MOSFET
5. 220v 60W incandescent bulbs
6. 12v supply for relay board, motors and Arduino
7. DHT11 (Digital temperature and humidity sensor) with 10k pullup
8. 802.11n Wifi Router and Ethernet cable
9. Wires, breadboard n stuff

Construction/Connections

The connections to various components are pretty straight-forward. The Arduino is connected to the 8 Channel relay board which turns the lights on or off.

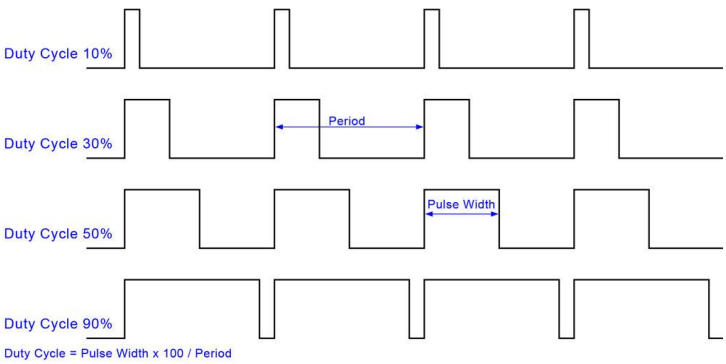
A relay is an electrically operated switch. That is, the state of the switch depends on the incoming signal. If the signal is HIGH, the relay is 'on' and the light glows. If the signal is LOW, the relay deactivates and the light is turned off.The speed of the FAN is controlled by PWM signals from the Arduino. The frequency of the PWM is about 490Hz. Since the relay used here is and electromechanical one, it can't be turned on or off very fast for the PWM to function properly. We need a purely electronic circuit for switching at high frequencies. Hence, a power MOSFET is used. A MOSFET is a type of transistor and they can easily be used to hundreds of kilohertz. Power MOSFETS are the ones with a higher power rating. The ones used here are P55 MOSFETS, rated up to 80Amps and 55 volts. The fans are connected to the Arduino via a MOSFET. Here, the MOSFET is used simply as a switch. As GATE voltage (V_{GS}) goes HIGH (>4v), the resistance between the terminals DRAIN and SOURCE become negligible, i.e. it acts like a closed switch. When the GATE voltage is low (<3.5v), the MOSFET deactivates and the resistance between the terminals is very large. Thus it acts as a closed switch.




220v AC
Living Room
Bedroom
Tubes
Incandescent
CFL
A/C
8-Ch Relay Board
ULN2803
12 v
P55
P55
12 v
12 v
DHT11
10k pull-up
ARDUINO UNO

PWM (Pulse Width Modulation)

PWM is a method used to vary the average power delivered to the load by switching it on and off for different intervals of time very fast. Eg, if we want to deliver 50% of the full power to a motor, we turn it on for 1ms and turn it off for 1ms. The amount of time it is 'on' is equal to the time it is 'off' hence, it receives only half the power in a larger time period. Also, the time it is 'off' or 'on' isn't really noticeable since this happens too fast for us to see. This way, appropriate amount of power can be delivered to the load.

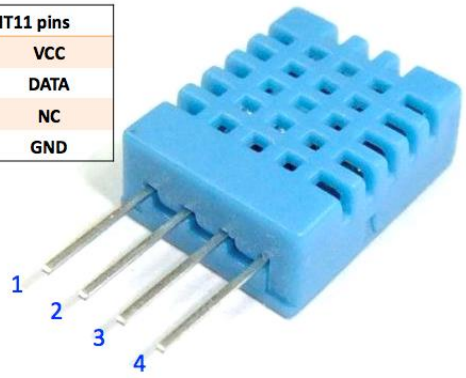


The percentage of the total time it is 'on' is usually called the duty cycle of the PWM wave.

$$\text{Duty Cycle} = \frac{Time\ the\ signal\ is\ HIGH}{Time\ the\ signal\ is\ HIGH\ +\ Time\ the\ signal\ is\ LOW}$$

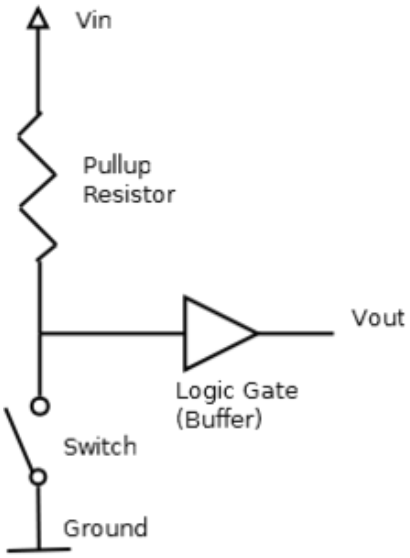
Functioning of the DHT11 (Digital temperature and humidity sensor)

DHT11 pins	
1	VCC
2	DATA
3	NC
4	GND



As the name suggests, the sensor is of digital kind. It does not output a linear voltage variation for the change in temperature/humidity, but send the data serially. It has 4 pins, of which one is not connected internally. The data pin is also required to be pulled up, ie connected to the Vcc with a 10k resistor.

A pull-up resistor weakly "pulls" the voltage of the wire it is connected to towards its voltage source level when the other components on the line are inactive. When all other connections on the line are inactive, they are high-impedance and act like they are disconnected. Since the other components act as though they are disconnected, the circuit acts as though it is disconnected, and the pull-up resistor brings the wire up to the high logic level. When another component on the line goes active, it will override the high logic level set by the pull-up resistor. The pull-up resistor ensures that the wire is at a defined logic level even if no active devices are connected to it.



Mechanism of working

When the Arduino boots, it first checks for the SD card and the webpage. If the SD card is accessible and webpage is present, it displays the message “SD init success” (init = initialization) over the serial interface. If the card is not present/accessible or if the SD card does not contain the webpage, the Arduino displays the message “SD init failed”

Once boot is completed, the Arduino is ready to receive requests for webpages.

When 192.168.0.20 is entered in the browser, the Ethernet shield responds with a “webpage request”. This request is detected by the Arduino. The Arduino sends a standard HTTP request header with a “connection: keep-alive” command. Next the Arduino reads the webpage stored in the SD card and sends it to the browser. The browser displays the page and sends a “GET” request for the background image file. As the request is received, the Arduino opens and reads the background file and sends it to the browser.

The data is sent in the form of unique “GET” requests. Inside these requests, strings of commands to be carried out are given. E.g., string like “&LED1=1”.

A GET request looks like:

Host: GET /ajax_inputs&nocache=206919.37673836946 HTTP/1.1

The request to turn on light 2 looks like this:

Host:GET /ajax_inputs&LED2=1&nocache=117936.32199987769 HTTP/1.1

```
SD init success

Page Request

GET / HTTP/1.1
Host: 192.168.0.20
Connection: keep-alive

BG request

GET /e.jpg HTTP/1.1
Host: 192.168.0.20
Connection: keep-aGET /ajax_inputs&nocache=430834.0318966657 HTTP/1.1
Host: GET /ajax_inputs&nocache=601181.4421508461 HTTP/1.1
Host: GET /ajax_inputs&nocache=501098.2546955347 HTTP/1.1
Host: GET /favicon.ico HTTP/1.1
Host: 192.168.0.20
Connection: GET /ajax_inputs&nocache=464125.2637375146 HTTP/1.1
Host: GET /ajax_inputs&nocache=846019.8587272316 HTTP/1.1
Host: GET /ajax_inputs&nocache=239076.40296965837 HTTP/1.1
Host:GET /ajax_inputs&nocache=848012.5048663467 HTTP/1.1
Host: GET /ajax_inputs&nocache=702262.7210244536 HTTP/1.1
Host: GET /ajax_inputs&nocache=546605.8801393956 HTTP/1.1
Host: GET /ajax_inputs&nocache=223855.20348325372 HTTP/1.1
Host:GET /ajax_inputs&nocache=498324.04125481844 HTTP/1.1
Host:GET /ajax_inputs&nocache=891797.3747011274 HTTP/1.1
```

The Arduino searches the command in the entire request. If the appropriate command is found, the action is carried out

Once the action has been carried out, the Arduino sends data to the client (your PC or web browser) as feedback/confirmation. AJAX is used to change parts of the page, like display temperature, humidity, change button states etc. Since the Arduino sends data to the client, all clients connected to 192.168.0.20 picks up the data and dynamically refreshes the page. This way, all browsers display the same information. That is, all browsers remain in sync.

Webpage contains JavaScript, AJAX, XML, HTML and CSS elements

1. JavaScript: Functions/actions to carry out when switches are clicked.
2. AJAX (Asynchronous JavaScipt and XML): Dynamically send/receive data and display them without refreshing the entire page.
3. XML (Extensible Markup Language): Store variables so that AJAX can access them.
4. HTML (Hyper Text Markup Language): Formatting of the page and text etc.
5. CSS (Cascading Style Sheets): Styling of buttons, boxes, background etc

Further possible improvements:

The concept of home automation is to automate/semi-automate certain tasks that are performed several times a day. It is no limited to just lights and fans. It can be further expanded to everything in the house. Some examples are:

- Timed water heaters: Water heaters use several kilowatts of energy. If left turned on all the time, it would waste a considerable amount of energy every day. If it can be timed however, the energy wastage is reduced substantially. E.g., the heater would turn on at 7:00am and would automatically turn off at 10:00pm. On weekdays, the time would be from 9:00 to 12:00. During winters, it can be programmed to be turned on for longer periods of time. Further implementation would be to control the temperature of the water too. E.g., it would maintain temperature from 40C to 60C only.

- Security systems: Wouldn't it be nice if we could lock you house from your phone remotely, just like we lock our fancy car with 'keyless entry'. Home automation can be implemented here too, but in a more advanced way. It can link security cams, (silent) burglar alarm, or just timed locking, so the house is automatically locked at 12:00 am and unlocked at 6:00am every day.

- Entertainment: Since our demo is a proof of concept for controlling lights and fans through wireless/wired LAN, it can be easily extended to electronic gadgets also. Internet connected TVs and HTPC (home theatre personal computer) are already available in the market today. It is possible to link them to the chain of automated gadgets. Its possible select which channel we want to watch and the TV would play it. Alternatively, the home theatre will directly play the music being played on the phone. Advanced implementation would go further and set the mood by dimming the lights for a room for a particular song or movie.

- Intelligent Lights: Wouldn't it be wonderful if the lights switch itself on/off by itself depending on the situation? Like, if no-one is in the room for more than 15 mins, all lights can be set to turn off. Or, depending on motion sensors, the lights may be programmed to turn on when someone is approaching the room. Some lights like the corridor or balcony lights may be set to turn on every evening and turn off every morning. Other lights may have some added functionality, like dimming or colour control.

- Power Failure situation: In a country like India, power failures are nothing out of the ordinary. Every now and then, power cuts occur and almost everyone is ready with an inverter as alternative power source. But usually, power saving techniques (switching to CFL instead of incandescent lights or fewer lights etc) are not applied or have to be applied manually, that result in a low battery. Intelligent systems may be programmed to
  - Turn off certain lights.
  - leave the higher priority lights on
  - Switch over to CFL automatically
  - Leave certain low power devices always on etc.

- A well-designed android app: For the sake of simplicity, we have used a webpage as an interface. But a standalone android app is better suited for practical purposes. It can have access to the home network over the internet when the user is out of local network range. So the user may turn on the AC or lift the garage door before even arriving.

- Automated garage doors: Just like security systems, the garage door can be automated to be controlled by the app/console.

- Voice control: Tell your TV to view a particular channel or tell your AC to cool things down a bit. It's the ultimate home automation.