**Home Automation**

**Making the Average Abode More Intelligent**

Awa Ka, Colin Redmond, Brian Curtis, Dean Nelson

Department of Information Technology and Management, Illinois Institute of Technology

Perlstein Hall 10 W. 33rd St., Room 223 Chicago, IL 60616

**ABSTRACT**

As time progresses and technologies improve, people are seeking more ways to do various tasks better and faster.  Using the projects outlined in this paper as examples, automation of menial tasks could become so easy as to become trivial.

**Categories and Subject Descriptors**

K.3.2 Computer and Information Science Education

**General Terms**

Algorithms

Management

Measurement

**Keywords**

home automation

arduino

**1. INTRODUCTION**

Technology has come so far. From the creation of computers to the automation of everyday tasks with computers, people are innovating day after day. When it comes to embedded Systems, they are becoming a big part of our lives as we are moving towards a technology driven lifestyle.

As automation is being applied in most areas, the area that interests us the most is home automation. People are constantly trying to improve their quality of life and the way by which daily operations are being completed.

There are so many areas to improve on when it comes to home automation. Therefore, our project will focus on four areas of home automation: light switch with app command, automatic laundry sensor, auto-irrigation system and home automation server.

**2. LIGHT SWITCH WITH ANDROID APP COMMAND**

Don’t you hate it when you get into your house or apartment and have to search for the light switch on the wall? Do you get lazy to get up and go turn off the light? The light switch is one of the most demanded and implemented devices during the automation process. It takes care of the issue of walking in the dark and bumping into objects, and walking on the carpet just to turn on the light. Right before arriving into the house, one can control all lights just by a click from the phone. The parts that

are needed to complete this project are: Arduino Uno, breadboard, relay, Bluetooth module, wires and android app.

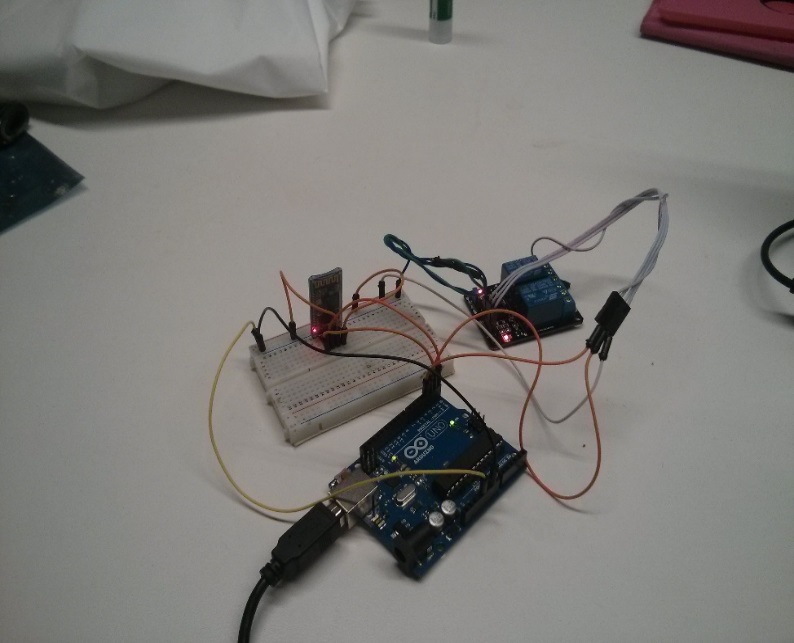
The Arduino Uno board is a microcontroller that we will be using in this

project as the main tool. It is composed of digital inputs/output pins, analog input pins, a ceramic resonator, a USB connection, a power jack, an ICSP header and a reset button. Some of these components will be relevant to this project.\The breadboard is used for prototyping the circuit layout of the project. As it does not need soldering, we will be using the wires to easily make changes on the design of the circuit prototype.

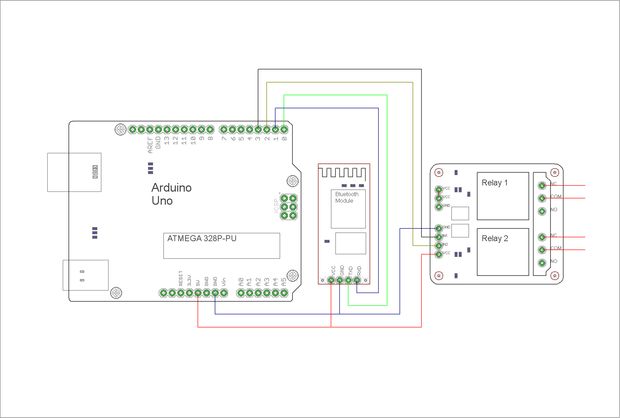
The relay is a 2-channel 5V relay module used for Arduino. We will be using it to control the Bluetooth module.

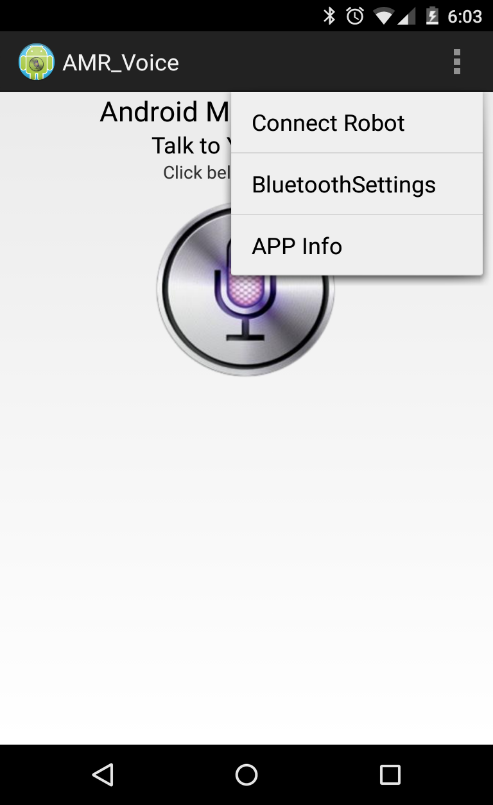
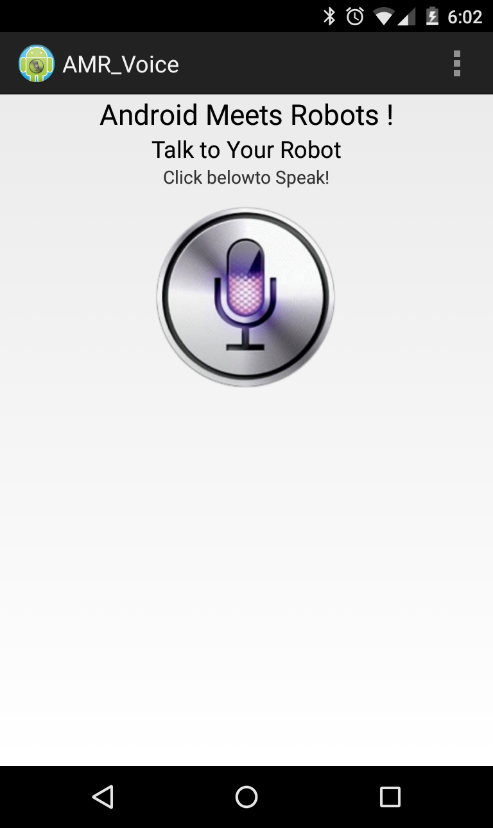
The Bluetooth module is the Arduino Wireless Bluetooth transceiver and it operates at 3.3V and can transmit up to 30 feet. It also has TXD (sender) and RXD (receiver) as communication ends to the Arduino.

The android app is an application called BT Voice Control for Arduino. It has been developed by SimpleLabs and it can be downloaded from the Google Playstore. This application will be used to send on/off commands to the relay. Below is an image of the completed prototype.



How does everything work? The Arduino sends power and voltage to the breadboard. It is also used to send data to the relay on pin 2 and 3. On the breadboard, the Bluetooth module is powered up and it also uses RXD and TXD to speak to the Arduino. Here is an image showing the diagram of the circuit layout.



Everything should light up when we plug the Arduino board to the computer. However the Bluetooth module should blink, which means it has not been paired up with the Android app yet. So the next step should be to open up the Android application and click on the 3-dot menu on the upper right corner and click on “Connect Robot”.

Then scan for devices to find the MAC address of the Bluetooth module. When the search is done, click on it and put the pass-code, which is usually 1234. Once connected, we can send “switch on” and “switch off” commands, which will open or close the relay (we should be able to hear the sound).

**3. AUTOMATIC LAUNDRY SENSOR**

This automatic laundry sensor was created to alert the user when exactly the laundry machine finishes the run and the load can be removed from the machine. This gadget will prevent the possibility of other people unloading the laundry when forgotten.

After considering this current issue, a simple one-way transmission and receiving system was created which, when combined with a local server, would be easy to hook up via Bluetooth, Wi-Fi, radio, etc. The next part was determining how the system would detect completion. The whole idea fell upon the “5 senses” rule; using sight, sound, taste, smell, and touch to sense when the laundry is done. “Taste” or “smell” would require sensors to maybe detect chemical levels of detergent or bacteria, which would have been invasive, complex, and possibly expensive. Sight was strongly considered for a while – using a laser (most likely infrared) to detect motion, or water pouring into the machine, but finding an IR laser proved to be difficult. Lastly, the two methods voted on were sound and touch – measuring the loudness of the machine, and the vibrations it gives off.

Sound was simple enough. A microphone is programmed to listen for noise input, and tell the processor “FALSE” if there was no noise detected. The threshold for noise was determined by doing several tests with a machine, giving a conclusion that “normal/quiet” noise registered a value of about 300, and a washing machine registered a value between 400 and 600, depending on the cycle.

Vibrations (movement) was a little trickier. This value was determined by the use of an accelerometer, which would register “movement” in values along the x, y, and z coordinate planes. Similar to the microphone, when the read values are within the normal parameters of its readings, it sends a FALSE value to the processor.

When the processor receives both FALSE values, it sends a signal to the server saying “Done” or whatever it has been programmed to do.

The parts used this complete this project are: Arduino MEGA, accelerometer, Electret microphone, mini breadboard, blue LED light, and five wires. The following code was used to program the Arduino board:

const int groundpin = A3; // analog input pin 4 -- ground

const int powerpin = A1; // analog input pin 5 -- voltage

const int xpin = A7; // x-axis of the accelerometer

const int ypin = A6; // y-axis

const int zpin = A5; // z-axis (only on 3-axis models)

const int microphonePin = A0; //the microphone positive terminal will connect to analog pin A0 to be read

const int ledPin = 13; //the code will flash the LED connected to pin 13

const int ledPin2 = 12; //the code will flash the LED connected to pin 12

int sample; //the variable that will hold the value read from the microphone each time

const int threshold = 400;//the microphone threshold sound level at which the LED will turn on

int scale = 200; //200 for 200g accelerometer - 3 for 3g version

int micsound; //variable if mic does NOT detect sound

int accmotion; //variable if mic does NOT detect motion

void setup(){

pinMode (ledPin, OUTPUT);//sets digital pin 13 as output

pinMode (ledPin2, OUTPUT);//sets digital pin 12 as output

pinMode(groundpin, OUTPUT);

pinMode(powerpin, OUTPUT);

digitalWrite(groundpin, LOW);

digitalWrite(powerpin, HIGH);

Serial.begin(9600);

}

void loop(){

{

sample = analogRead(microphonePin); //the arduino takes continuous readings from the microphone

Serial.print ("start timer");

delay (30000);

Serial.println();

Serial.print ("End timer");

Serial.println();

}

{

if (sample > threshold) {

digitalWrite (ledPin, HIGH); //if the reading is greater than the threshold value, LED turns on

delay (1000); //LED stays on for a half a second

digitalWrite (ledPin, LOW); //LED turns off

micsound = true;

}

else

digitalWrite (ledPin, LOW);

micsound = false;

}

{

// print the sensor values:

Serial.print("X: "); Serial.print(analogRead(xpin));

// print a tab between values:

Serial.print("\t");

Serial.print("Y: "); Serial.print(analogRead(ypin));

// print a tab between values:

Serial.print("\t");

Serial.print("Z: "); Serial.print(analogRead(zpin));

Serial.println();

// delay before next reading:

delay(500);

}

{

if (analogRead(A7) == 340 || analogRead(A7) == 341 && analogRead(A6) == 341 && analogRead(A5) == 343 || analogRead(A5) == 344){

accmotion = false;

}

else

accmotion = true;

}

{

if (micsound == false && accmotion == false){

digitalWrite (ledPin, HIGH);

Serial.print ("Done!");

while (ledPin, HIGH);

}

else

digitalWrite (ledPin, LOW);

}

}

**4. Automated Irrigation System**

There is just no comparison between the taste of fruits fresh from the vine, and the fruit bought from the store, no matter how local or organic the supplier is, homegrown always tastes better, Yet so many people just do not have the time necessary to grow their own food. For others, the constant bending down to water the plants wears on their back. This project fixes all the problems.

What used to be a daily chore to water the garden turn into a once a month after thought to fill up the water tank. Freeing up time for other pursuits, or enabling those who otherwise would not have the time to be able to grow some of their own food. Giving them easy and quick access to fresh fruits and vegetables with minimal work. Giving them the ability to grab a handful of grape tomatoes in the morning, instead of that high sugar breakfast bar.

Others are prevented by physical limitations, for they are unable to bend down for the long periods of time that are necessary to water plants. With this project, they don’t have to. All they have to do to fill the tank is place the hose in the input pipe, and turn the hose off when the tank is full. No bending over required.

One of the goals of this project was to maintain ease of scalability, This project utilizes one Arduino Uno, a humidity sensor, a relay, a water tank, a pump, tubing, and some pvp pipe. to increase the growing capacity of the project, just add one, relay, pump, and humidity sensor, for each growing area.

The operations of the project are fairly simple. The humidity sensor takes readings from the soil so that when the soil humidity drops beneath the threshold, the arduino then automatically turns on the relay which triggers the pump. The pump is then run in controlled doses to prevent overwatering and drowning the plants. Giving the plants the optimum amount of water they need to grow, so they can provide the most food.

Included below is the code used to operate the project.

**#include <Sensirion.h>**

**const int Relay = 7;**

**const uint8\_t dataPin = 9; // SHT serial data**

**const uint8\_t sclkPin = 8; // SHT serial clock**

**const uint8\_t ledPin = 13; // Arduino built-in LED**

**const uint32\_t TRHSTEP = 5000UL; // Sensor query period**

**const uint32\_t BLINKSTEP = 250UL; // LED blink period**

**Sensirion sht = Sensirion(dataPin, sclkPin);**

**uint16\_t rawData;**

**float temperature;**

**float humidity;**

**float dewpoint;**

**byte measActive = false;**

**byte measType = TEMP;**

**unsigned long trhMillis = 0; // Time interval tracking**

**unsigned long blinkMillis = 0;**

**void setup() {**

**Serial.begin(9600);**

**pinMode(ledPin, OUTPUT);**

**pinMode(Relay, OUTPUT);**

**delay(15); // Wait >= 11 ms before first cmd**

**// Demonstrate blocking calls**

**sht.measTemp(&rawData); // sht.meas(TEMP, &rawData, BLOCK)**

**temperature = sht.calcTemp(rawData);**

**sht.measHumi(&rawData); // sht.meas(HUMI, &rawData, BLOCK)**

**humidity = sht.calcHumi(rawData, temperature);**

**dewpoint = sht.calcDewpoint(humidity, temperature);**

**logData();**

**}**

**void loop() {**

**unsigned long curMillis = millis(); // Get current time**

**// Rapidly blink LED. Blocking calls take too long to allow this.**

**if (curMillis - blinkMillis >= BLINKSTEP) { // Time to toggle the LED state?**

**ledState ^= 1;**

**digitalWrite(ledPin, ledState);**

**blinkMillis = curMillis;**

**}**

**// Demonstrate non-blocking calls**

**if (curMillis - trhMillis >= TRHSTEP) { // Time for new measurements?**

**measActive = true;**

**measType = TEMP;**

**sht.meas(TEMP, &rawData, NONBLOCK); // Start temp measurement**

**trhMillis = curMillis;**

**}**

**if (measActive && sht.measRdy()) { // Note: no error checking**

**if (measType == TEMP) { // Process temp or humi?**

**measType = HUMI;**

**temperature = sht.calcTemp(rawData); // Convert raw sensor data**

**sht.meas(HUMI, &rawData, NONBLOCK); // Start humidity measurement**

**} else {**

**measActive = false;**

**humidity = sht.calcHumi(rawData, temperature); // Convert raw sensor data**

**dewpoint = sht.calcDewpoint(humidity, temperature);**

**logData();**

**Water();**

**}**

**}**

**}**

**void logData() {**

**Serial.print("Temperature = "); Serial.print(temperature);**

**Serial.print(" C, Humidity = "); Serial.print(humidity);**

**Serial.print(" %, Dewpoint = "); Serial.print(dewpoint);**

**Serial.println(" C");**

**}**

**void Water() {**

**if (humidity < 70.0) {**

**digitalWrite(Relay, HIGH);**

**Serial.println("Relay ON");**

**delay(2000);**

**digitalWrite(Relay, LOW);**

**Serial.println("Relay OFF");**

**} else {**

**digitalWrite(Relay, HIGH);**

**Serial.println("Relay OFF");**

**}**

**delay(2000);**

**}**

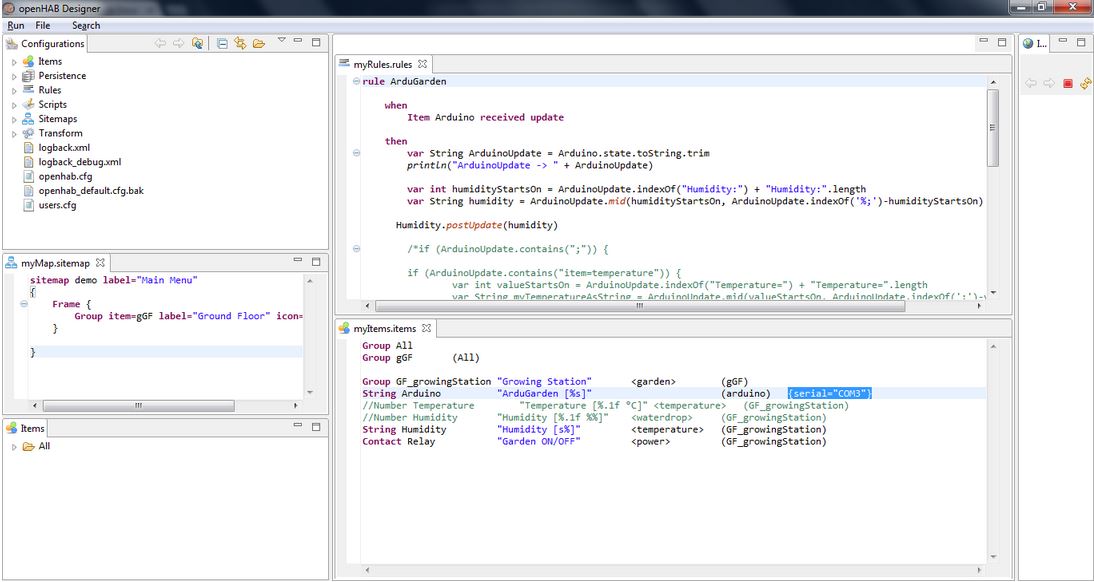
**5. Home Automation Server**

Of course, it would be nice to be able to see the information from all the various sensors in one place right? With a home automation server, one can gather data and device states from multiple places and view the data in one convenient place (in this case, a personal website).

The server software used in this project is an open-source project called OpenHab. OpenHab is a very versatile and powerful java-based home automation tool that can gather data from a variety of different source types. It accomplishes this through “bindings” which correlate items with devices.

OpenHab is configured with three files: the .Items file, the .Rules file, and the .Sitemap file. The .Items file stores items which are comprised of a variable type, variable, text format, sitemap icon, groups, and a binding e.g. Number Humidity "Humidity [%.1f %%]" <waterdrop> (Gardens) {serial="COM1"}. The items are then processed by the rules engine dictated by the .Rules file which consists of java code. Using this code, one can tell OpenHab what to do with the information it receives such as IF (Humidity < 60){Relay = ON}. The information is then displayed on the sitemap. The following images are a representation of the setup and the results that should be displayed.





# 6. REFERENCES

["1"] Arduino Home Automation (Bluetooth).

(n.d.). Retrieved December 5, 2014, from <http://www.instructables.com/id/Arduino-Home-Automation-Bluetooth/?ALLSTEPS>

# [“2”]Kreuzer, K. openhab/openhab wiki, 2014. Retrieved December 4, 2014, from GitHub: <https://github.com/openhab/openhab/wiki>

[“3”] <http://playground.arduino.cc/Code/Sensirion>