

**DESIGN OF A HOME AUTOMATION SYSTEM USING ARDUINO WITH
WIRELESS CONTROL**

By

AZRUL REZUAN BIN AZMAN

FINAL PROJECT REPORT

Submitted to the Department of Electrical & Electronic Engineering
in Partial Fulfillment of the Requirements
for the Degree
Bachelor of Engineering (Hons)
(Electrical & Electronic Engineering)

Universiti Teknologi PETRONAS

Bandar Seri Iskandar

31750 Tronoh

Perak Darul Ridzuan

© Copyright 2013

by

Azrul Rezuan Bin Azman, 2013

CERTIFICATION OF APPROVAL

DESIGN OF A HOME AUTOMATION SYSTEM USING ARDUINO WITH WIRELESS CONTROL

by

Azrul Rezuan Bin Azman

A project dissertation submitted to the
Department of Electrical & Electronic Engineering
Universiti Teknologi PETRONAS
in partial fulfilment of the requirement for the
Bachelor of Engineering (Hons)
(Electrical & Electronic Engineering)

Approved:

(Mohd Azman Bin Zakariya)
Project Supervisor

UNIVERSITI TEKNOLOGI PETRONAS
TRONOH, PERAK

May 2013

CERTIFICATION OF ORIGINALITY

This is to certify that I am responsible for the work submitted in this project, that the original work is my own except as specified in the references and acknowledgements, and that the original work contained herein have not been undertaken or done by unspecified sources or persons.

Azrul Rezuan Bin Azman

ABSTRACT

The rapid growth of technology has drastically changed the living standards of modern society. Seeing the increasing number of electronic devices being made in a household, an automated home control system has become an increasingly useful feature. Current systems, however, have problems with complexity, high costs, non-open sources and multiple incompatible standards; resulting in the limited venture of the home automation into the homes of the rich or hobbyists.

This project intends to design an open source, affordable and easy to use home automation system, which will be done by interfacing the open source Arduino microcontroller with web browser; creating a simple, easy-to-use system to control home appliances.

The project was carried out in several stages. The Arduino is first setup for Wi-Fi connection and web browser interfacing. Next, the system is made to work with appliances in a home model. Last of all, a simple user interface is created using HTML to make the system user friendly, completing the home automation system design.

The home automation system design was successfully implemented. Using web browser on laptop and smartphone, connection was made to Arduino through wireless network, allowing for control of the appliances in the home model.

ACKNOWLEDGEMENTS

All praises be to God, for with his help and guidance, I was able to complete my final year project and write this dissertation with success.

I would like to offer my deepest gratitude to my supervisor, Mr Mohd Azman Bin Zakariya, Lecturer of Electrical & Electronics Department, UTP. His supervision and continuous support helped greatly during the past two semesters in completing my final year project. He provided me with guides, sample codes, and helped me to understand concepts. He also helped in correcting my documents with care.

Finally, I would like to great appreciation to all my other lecturers, colleagues, family, friends and everyone who supported me for the past two semesters, thus making my final year project a great success.

TABLE OF CONTENTS

ABSTRACT.....	iv
ACKNOWLEDGEMENTS.....	v
LIST OF FIGURES	viii
LIST OF TABLES	ix
CHAPTER 1 INTRODUCTION.....	1
1.1 Background of Study.....	1
1.2 Problem Statement	1
1.2.1 Problem Identification	1
1.2.2 Project Significance	2
1.3 Objective and Scope of Study	2
1.3.1 Objective	2
1.3.2 Scope of Study	2
1.4 Relevancy of the Project	2
1.5 Feasibility of the Project	3
CHAPTER 2 LITERATURE REVIEW.....	4
2.1 A Review of Existing Home Automation Systems.....	4
2.2 Arduino.....	5
2.2.1 Terminology.....	5
2.2.2 Arduino Uno	5
2.2.3 Inside the Arduino Uno.....	8
2.2.4 Programming the Arduino	12
2.3 Arduino WiFi Shield	16
2.4 Wireless Network.....	19
2.4.1 802.11b.....	19
2.4.2 802.11g.....	19
2.5 Relays	20
2.5.1 Relay Configurations	20
CHAPTER 3 METHODOLOGY.....	22
3.1 Research Methodology.....	22
3.2 Gantt Chart	24
3.2.1 Final Year Project I Milestones:	24

3.2.2	Final Year Project II Key Milestones:	25
3.3	Tools.....	25
3.3.1	Hardware.....	25
3.3.2	Software	25
CHAPTER 4	RESULTS AND DISCUSSION	26
4.1	System Setup	26
4.2	System Operation	27
4.3	Program	31
4.3.1	Setup and Server Connection.....	31
4.3.2	Data processing.....	31
CHAPTER 5	CONCLUSION AND RECOMMENDATIONS.....	34
CHAPTER 6	REFERENCES	35
APPENDICES	36
APPENDIX A	ARDUINO PROGRAM CODE.....	37

LIST OF FIGURES

Figure 1: Block diagram of an Arduino I/O Board [5]	5
Figure 2: Layout of the Arduino Uno [8].....	6
Figure 3: Arduino Uno Schematic [6]	7
Figure 4: Simplified block diagram of the ATmega328 [5]	8
Figure 5: Systems available in the Arduino Uno [8]	9
Figure 6: Arduino Uno Expansion Connectors [5]	10
Figure 7: An example of connections between Ethernet Shield and Arduino board when stacked together [9]	12
Figure 8: Programming the Arduino Uno [8]	13
Figure 9: Arduino Development Environment Interface, showing the example sketch 'Blink' [5]	14
Figure 10: Arduino Development Environment buttons [8].....	14
Figure 11: Summary of Arduino Development Environment built-in features [8] ...	16
Figure 12: Arduino WiFi Shield: Front and Back [10].....	16
Figure 13: Arduino WiFi Shield stacked on top of Arduino board [9].....	17
Figure 14: Pins in use on the Arduino WiFi Shield [10]	17
Figure 15: Relay Configurations [12]	21
Figure 16: Research Methodology Process Flow Chart.....	22
Figure 17: Overall System Design	23
Figure 18: Circuit Diagram for the Home Automation System. The LED represents the lighting control, while the two motors represent ventilation and security control	26
Figure 19: Serial monitor showing the Arduino's activity. The IP address is boxed in orange.....	27
Figure 20: Sending an HTTP request to the Arduino using Firefox web browser. In this case an ON command for the light bulb is being sent to the Arduino	28
Figure 21: Serial monitor showing the HTTP request received by the Arduino	28
Figure 22: After receiving command light bulb is switched ON.....	29
Figure 23: Web browser showing the HTTP response sent by the Arduino	29
Figure 24: Block diagram representing system setup and operation	30
Figure 25: Arduino Program Process Flowchart: Setup and Server Connection.....	32
Figure 26: Arduino Program Process Flowchart: Data processing.....	33

LIST OF TABLES

Table 1: Arduino Expansion Connector Pin Names [5]	11
Table 2-3: Summary of WLAN Standards used	19
Table 3-1: Final Year Project I Gantt Chart.....	24
Table 3-2: Final Year Project II Gantt Chart	24

CHAPTER 1

INTRODUCTION

1.1 Background of Study

The rapid growth of technology has drastically changed the living standards of modern society. Homes now have ready available access to electric power, radio, televisions, and telephones. Domestic chores which were previously a laborious process are now made easier by specialized machines. For instance, washing and drying machines reduce the labor of cleaning and drying clothes. Water heaters on the other hand have made it easier for bathing at any time, while the task of food preservation has totally been handed over to refrigerators. Heating, ventilation and air conditioning (HVAC) appliances provide further improved convenience and comfort.

Seeing the various numbers of electronic devices being made available in a household, a remote, centralized, automated home control system has become an increasingly useful and desired feature. A home automation system allows electrical appliances to interconnect with one another, allowing for better convenience, energy efficiency, and security.

1.2 Problem Statement

1.2.1 Problem Identification

Home automation has been a feature of science fiction for many years, and began to be put into practice in the early 20th century. Despite the great interest, however, problems have limited the venture of home automation into the homes of the rich or hobbyists, among them including complexity, high costs, and multiple incompatible standards.

1.2.2 Project Significance

This project intends to encourage the use of home automation systems in all classes of homes, through the design of a low cost, easy to use, and open source home automation system; ultimately helping in providing a convenient, energy efficient, and secure environment for the society.

1.3 Objective and Scope of Study

1.3.1 Objective

This project focuses on the design of a home automation system as an open source platform for all end users. The objectives of the project are as follows:

- To design and construct a microcontroller-based system that effectively controls and monitors devices in the home system
- To develop a remote control system that enables data transfer through wireless transfer medium
- To interface the system with a web browser enabling cross platform control
- To design a system that is user friendly and easy to use

1.3.2 Scope of Study

The aim of this project is to design an open source, easy-to-use and affordable home automation system. For this reason the Arduino microcontroller is used, acting as the main controller for the system by sending signals to control electronic appliances. The project will be limited to a home model for prototyping purposes. A web browser works as the user interface while a standard wireless network is used as the medium between Arduino and web browser.

1.4 Relevancy of the Project

This project is a result from analyzing existing home automation systems which were found to be non-open source, complex and high costing for implementation. *Design of a Home Automation System Using Arduino with Wireless Web Browser Control* offers average users a low cost, easy to use, and open source platform for employing

home automation into their homes allowing for better energy saving, convenience and efficiency.

1.5 Feasibility of the Project

The project is executed within a two semester frame. The time frame allocated for this project is limited but adequate through proper time management and planning. As the project only involves a prototype, the costs are estimated to be within the given budget.

CHAPTER 2

LITERATURE REVIEW

2.1 A Review of Existing Home Automation Systems

Different types of approaches have been made towards home automation. An SMS based method uses GSM technology available in phones to communicate with a microcontroller which acts as the main control for access to home appliances. A GSM module is also required to be attached to the microcontroller through a port to enable SMS capability [1]. The disadvantage of such a system is that it is not user friendly, as there is no graphical user interface, and access codes and command codes must be remembered to operate the system [2].

Another approach focuses on voice recognition to send commands through a wireless RF network. The voice command is captured using a microphone, digitalized, and sent to a computer to be processed by a program based on Visual Basic which employs Microsoft speech API. Upon recognition of the voice command, control signals are sent to the specified appliance addresses for action. The tested system however was not always accurate in recognizing voice commands [3].

Hand gestures were also proposed as control for home automation systems by [4]. A small camera is worn as a necklace to observe the various gestures made by a user's hand in order to interpret and send command signals. The use of such technology, however, requires the use of a high end PC for data processing, resulting in a higher setup cost.

2.2 Arduino

2.2.1 Terminology

The term *Arduino* covers the hardware, software, development team, design philosophy, and morale of the user community. Originally developed in Ivrea, Italy, Arduino was named after the king of Italy about 1000 years ago, “Arduin of Ivrea”. The name *Arduino* is a masculine Italian name meaning “strong friend”, and is always capitalized being a proper name [5].

The Arduino I/O Board is the physical, tangible part of the Arduino system. The board is based on the Atmel AVR ATmega8 microprocessor and later derivatives containing a serial port, power supply circuitry, expansion connectors, and various support components [5].

Figure 1 depicts the block diagram of the Arduino board:

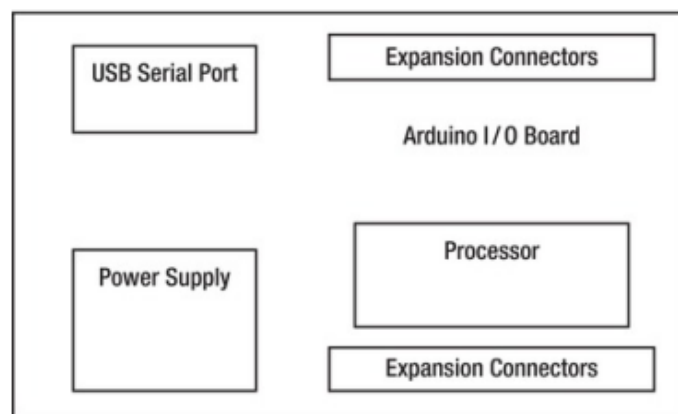


Figure 1: Block diagram of an Arduino I/O Board [5]

There are several different models of the Arduino I/O Board. The board portrayed in

Figure 1 above is Arduino Uno, which will be further discussed in the next section.

2.2.2 Arduino Uno

For this project the Arduino UNO, a board based on the ATmega328 microcontroller, is used. The board features 14 digital I/O pins, 6 analog inputs, 16 MHz ceramic

resonator, a USB connection, a power jack, an ICSP header, and a reset button [6]. For the board's programming software, an IDE development environment and core libraries are involved. The IDE is written in Java, while the code libraries are written in C and C++ [7].

Figure 2 illustrates the layout of the Arduino Uno:

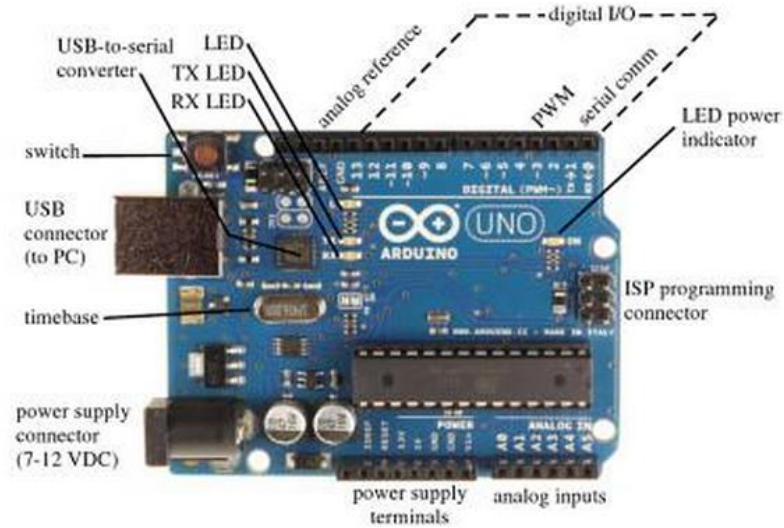


Figure 2: Layout of the Arduino Uno [8]

2.2.3 Inside the Arduino Uno

Processor – Atmel ATmega328

The Atmel ATmega328 is the microcontroller which functions as the brains of the Arduino Uno, containing a central processing unit (CPU), memory arrays, clocks, and peripherals; basically a computer on a chip. A simplified block diagram of the ATmega328 can be seen below:

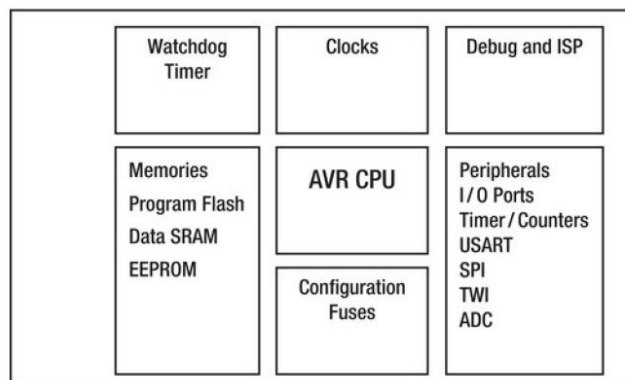


Figure 4: Simplified block diagram of the ATmega328 [5]

ATmega328 can operate from 1.8 to 5.5V, making it suitable for battery-powered applications. Lower voltages however, have a lower maximum clock rate. To run at the maximum rated clock rate of 20MHz, at least 4.5V supply is required. As the Arduino I/O board supplies 5.0V to the ATmega328, the processor can run at any speed up to the maximum of 20MHz [5].

The ATmega328 comes with a wide variety of features such as:

- Memory system,
- Port system,
- Timer system,
- Analog-digital-converter (ADC),
- and Serial communications.

A simple chart of the systems available in Arduino can be seen in Figure 5:

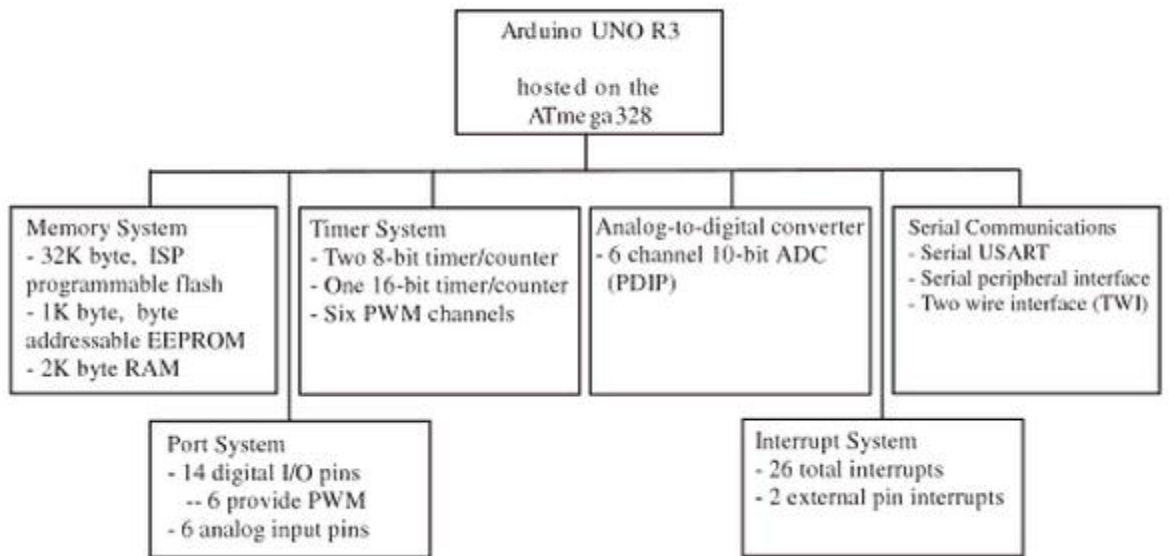


Figure 5: Systems available in the Arduino Uno [8]

Serial Port

The serial port is used to communicate between the Arduino and PC in the development stage when uploading programs to the I/O Board. There are several types of serial communication protocols. The Arduino's serial port is used in an asynchronous mode, meaning it doesn't require an independent clock signal. The asynchronous method uses one signal to transmit data and another to receive data [5].

Power Supply

There are a number of ways to power an Arduino, the simplest method having the Arduino's USB cable connected to a PC. The USB standard allows for the supply of up to 100mA for an unenumerated USB device and as much as 500mA for a properly enumerated USB device, which is enough to power several LEDs and low-power sensors. For greater electrical loads, however, an external, stronger power supply will be required [5].

Expansion Connectors/Ports

The Arduino Uno provides four sets of expansion connectors with the purpose of adding additional circuitry:

- Power connector
- Analog connector
- Digital I/O connectors (2 sets)

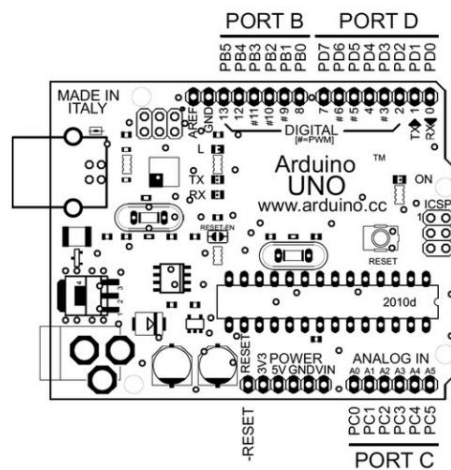


Figure 6: Arduino Uno Expansion Connectors [5]

The power connector provides connection to the main supply voltages (Vin, 5V, 3V3, GND) as well as the -RESET pin. The analog connector presents the six analog inputs, A0-A6, which can also be configured to be used as digital I/O lines [5].

In the ATmega328 convention, these digital I/O connectors are called ports, namely Port B, Port C, and Port D. Each of the I/O pin has an alternate peripheral function, configurable when programming the Arduino [5]. : Arduino Expansion Connector Pin Names below shows the list of all the expansion connector pin information along with the respective pins on the AVR ATmega328:

Connector	Arduino	AVR
J1/IOL	D0/RX	PD0/RXD
	D1/TX	PD1/TXD
	D2	PD2/INT0
	D3/PWM	PD3/INT1/OC2B
	D4	PD4
	D5/PWM	PD5/OC0B
	D6/PWM	PD6/OC0A
J3/IOH	D7	PD7
	D8	PB0
	D9/PWM	PB1/OC1A
	D10/PWM	PB2/OC1B
	D11/PWM	PB3/OC2A
	D12	PB4
	D13/LED	PB5

Connector	Arduino	AVR
J2/AD	A0/D14	PC0/ADC0
	A1/D15	PC1/ADC1
	A2/D16	PC2/ADC2
	A3/D17	PC3/ADC3
	A4/D18	PC4/ADC4/SDA
	A5/D19	PC5/ADC5/SCL

Table 1: Arduino Expansion Connector Pin Names [5]

Shields

Additional features and external hardware may be added to the Arduino using the daughter card concept, known as “Shields” in the Arduino convention. These shields mate with the expansion connectors on the arduino board, allowing the I/O Board to act like a small motherboard, providing mechanical and electrical connections to additional circuitry. Focus will be given to the the Arduino WiFi Shield being one of the main components in this project, and will be further elaborated in the next section .

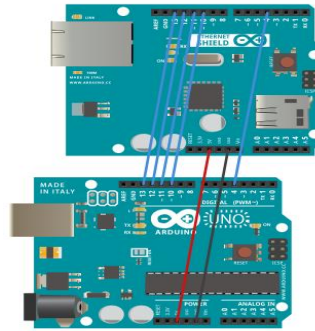


Figure 7: An example of connections between Ethernet Shield and Arduino board when stacked together [9]

2.2.4 *Programming the Arduino*

This section will give an overview of how the Arduino Uno is programmed. A compiler, hosted on a computer separate from the Arduino Uno, produces a machine code to be uploaded into the Arduino board (.hex file), using the program source files provided by the program writer (.c and .h files). This is done in three steps.

1. Compilation – program source files are converted into assembly code (.asm file).
2. Assembly language file is then delivered to assembler which then transforms it into machine code (.hex file) suitable to be uploaded to the Arduino Uno.
3. Program is uploaded into Arduino Uno.

An illustration of the process is given in Figure 8 below:

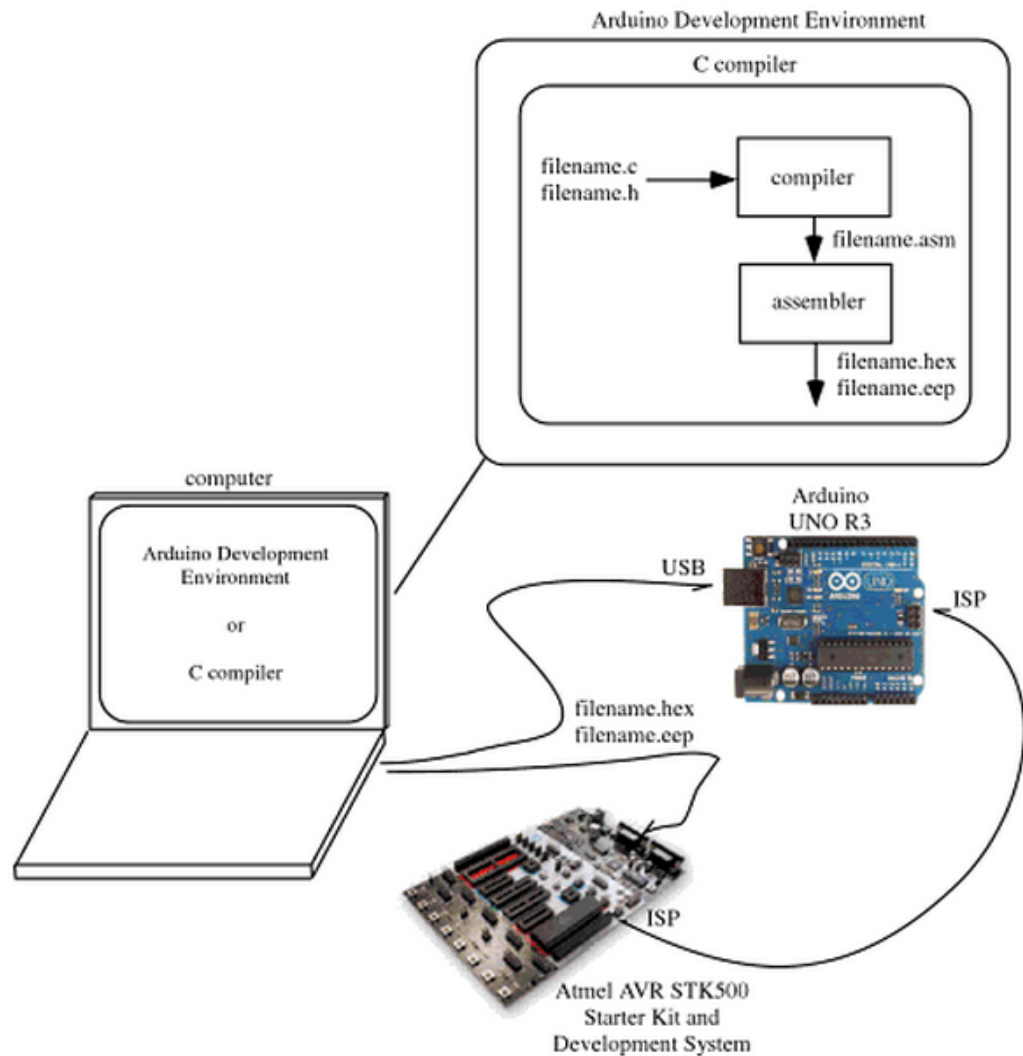


Figure 8: Programming the Arduino Uno [8]

The Arduino Development Environment (ADE) provides users with a friendly interface making the development and programming process easier. The ADE will be discussed in more detail in the next subsection.

Arduino Development Environment

The Arduino Development Environment contains a text editor, a message area for status displays, a text console, a tool bar of common functions, and a menu bar. ADE also provides a user friendly interface allowing for quick code upload, which is possible as the Arduino Uno is ready-made with a bootloader program [8].

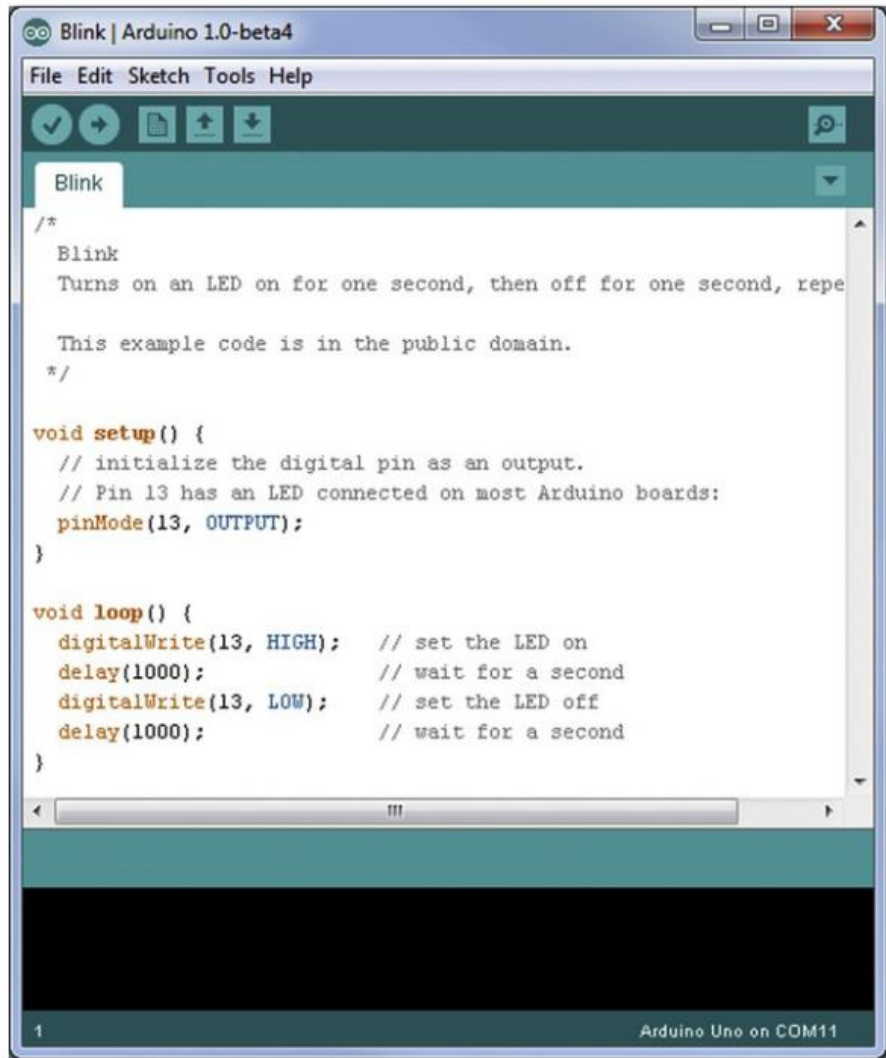


Figure 9: Arduino Development Environment Interface, showing the example sketch 'Blink' [5]

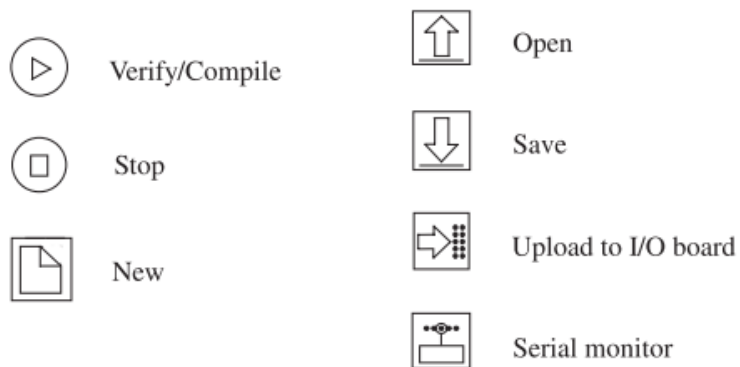


Figure 10: Arduino Development Environment buttons [8]

Figure 10 above shows the buttons available on the ADE toolbar. The buttons provide single click access to the most commonly used features. “Upload to I/O Board” button compiles written code and uploads it to the Arduino Board, while “Serial Monitor” button opens the serial monitor feature, allowing text data to be send to and received from the Arduino board. The other buttons are self-explanatory [8].

Sketchbook Concept

In order to make the Arduino a friendly platform for art students, the sketchbook concept is employed in the Arduino Environment. Programs written are called ‘sketches’ and maintained within a sketchbook in the Arduino environment. An individual sketch can be accessed via the ‘Sketchbook’ entry under the file tab [8].

Built-in Functions

The Arduino Development Environment contains a number of built-in functions that allow users to construct sketches quickly. The functions are summarized in Figure 11.

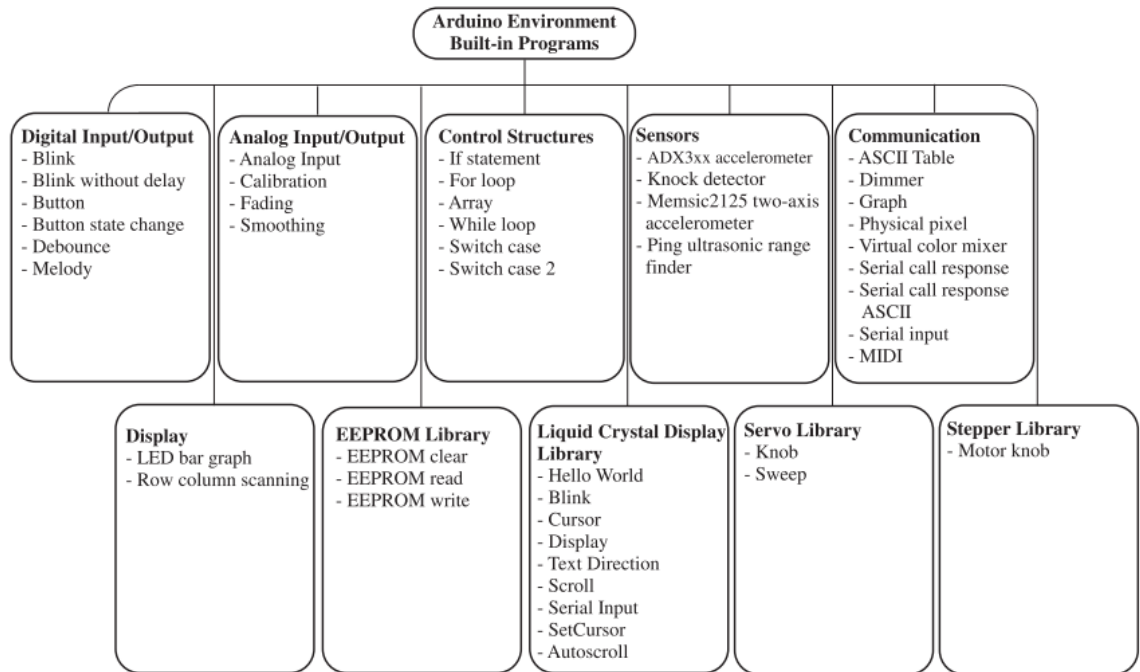


Figure 11: Summary of Arduino Development Environment built-in features [8]

2.3 Arduino WiFi Shield

The Arduino WiFi Shield is based on the HDG104 Wireless LAN 802.11b/g System in-Package (SiP), allowing an Arduino board to connect to the internet using the 802.11b and 802.11g wireless specifications (WiFi). The WiFi library is used to write sketches (programs) to connect to the internet using the shield. An onboard micro-SD card slot accessible through the SD Library is also available and can be used to store files for serving over the network [10]. Figure 12 below shows an Arduino WiFi Shield.

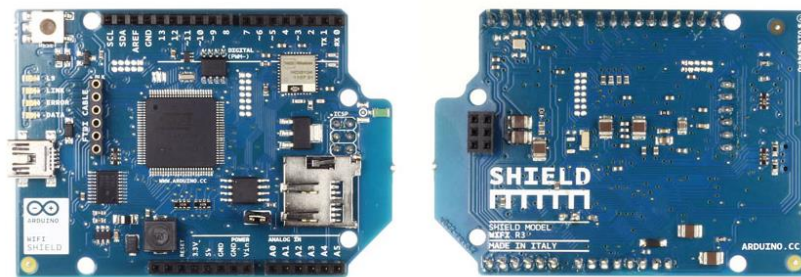


Figure 12: Arduino WiFi Shield: Front and Back [10]

The Arduino WiFi Shield can be connected by stacking the shield on top of the Arduino board as shown in Figure 13: Arduino WiFi Shield stacked on top of Arduino board below.

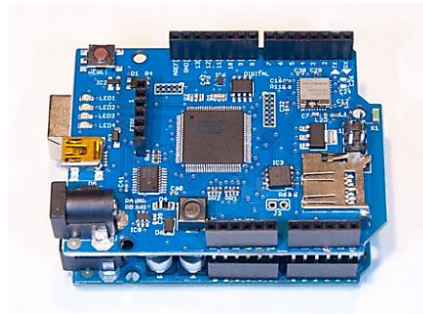


Figure 13: Arduino WiFi Shield stacked on top of Arduino board [9]

The Arduino WiFi Shield communicates with the Arduino using the SPI (serial peripheral interface) bus through the ICSP header, located on pins 11, 12, and 13 on the Arduino Uno. Pin 10 is used to select the HDG104 and pin 4 is used for the SD card. Additionally, pin 7 is used as a handshake pin between the WiFi shield and the Arduino. As a result, each of these pins cannot be used for general I/O [10]. Figure 14 below demonstrates each of the pins in use.

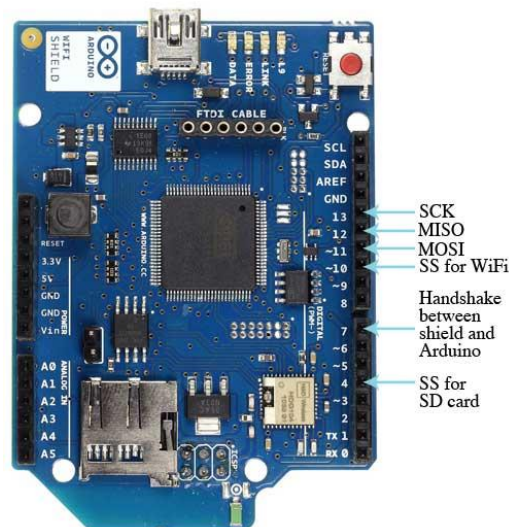


Figure 14: Pins in use on the Arduino WiFi Shield [10]

Connecting to a network with the WiFi shield requires the name of the network, known as SSID (service set identifier) and a password or key depending on the type of connection. The WiFi shield can connect to open networks or secured networks

with WEP, WPA, or WPA2 encryption. A password is required for WPA and WPA2 networks, while for WEP a key and key index is necessary [9]. Configuration for the network setup can be seen below:

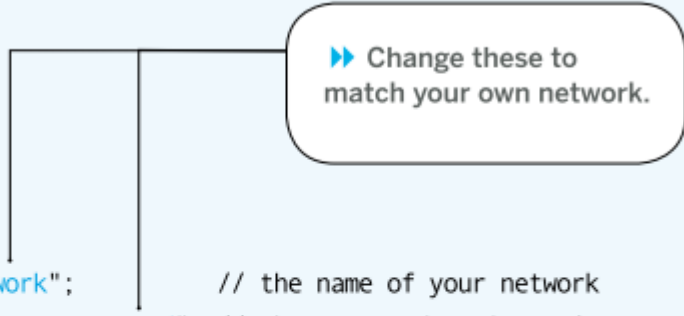
Configuration for WPA [9]

```
/*
  WiFi RGB Web Server
  Context: Arduino
  */

#include <SPI.h>
#include <WiFi.h>

char ssid[] = "myNetwork";           // the name of your network
char password[] = "secretpassword"; // the password you're using to connect
int status = WL_IDLE_STATUS;        // the WiFi radio's status

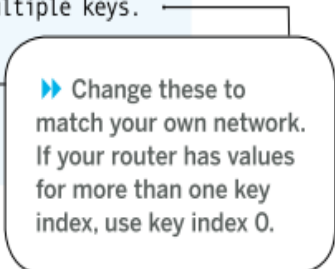
Server server(80);
int lineLength = 0; // length of the incoming text line
```



Change these to match your own network.

Configuration for WEP [9]

```
char ssid[] = "myNetwork";           // the name of your network
char keyIndex = 0;                    // WEP networks can have multiple keys.
// the 128-bit WEP key you're using to connect:
char key[] = "FACEDEEDDADA01234567890ABC";
int status = WL_IDLE_STATUS;        // the WiFi radio's status
```



Change these to match your own network. If your router has values for more than one key index, use key index 0.

2.4 Wireless Network

Wireless networks or WLANS (wireless local area networks) are networks which transmit signals through the atmosphere via radio frequency waves. IEEE's 802.11 committee develops wireless standards which are the most popularly used today. Two of the wireless standards are used in this project; namely the 802.11b and 802.11g [11].

2.4.1 802.11b

The first 802.11 standard which took hold in 1999, 802.11b uses a frequency range of 2.4 to 2.4835 GHz (2.4GHz band), providing a theoretical maximum of 11Mbps throughput, which is typically around 5Mbps in actual throughput. A radius of up to 100 meters (approximately 330 feet) between the wireless nodes and access points or other wireless nodes is required to achieve the specified throughput. 802.11b is the least expensive of all 802.11 WLAN technologies [11].

2.4.2 802.11g

A newer standard released by the 802.11, the 802.11g is designed to be just as affordable as 802.11b while increasing the maximum theoretical throughput to 54Mbps, which ranges generally between 20 to 25 Mbps in effective throughput. The frequency range and geographic range on the other hand is still the same as 802.11b, using the 2.4GHz frequency band with a geographic range of 100 meters (approximately 300 feet). 802.11g is backward compatible with 802.11b [11].

Standard	Frequency Range	Theoretical maximum throughput	Effective throughput (approximate)	Average geographic range
802.11b	2.4 GHz	11 Mbps	5 Mbps	100 meters
802.11g	2.4 GHz	54 Mbps	20-25 Mbps	100 meters

Table 2 Summary of WLAN Standards used

2.5 Relays

A relay is a type of electromechanical switch. When a current is applied across a relay's control pins, the control pins activate an electromagnet that attracts a movable lever, activating the switch. As the control pins and switch are electrically separated, a relay can be used to control large current with a small current [12]. This means that it can be used as a switch to control high-power loads such as AC/DC lighting and electronic appliances, with input from the Arduino which outputs about 40mA [5].

Hence, for this project, relays are most suitable to be used as switches to electronic appliances as they can be activated by the low current signals from Arduino.

2.5.1 Relay Configurations

There are various configurations for a relay's contacts depending on its use. Four common types of relays will be discussed here [12]:

1. Single Pole, Single Throw (SPST): This type of relay uses one coil to control one switch with two contacts
2. Single Pole, Double Throw (SPDT): This type of relay uses one coil to operate one switch with three contacts.
3. Double Pole, Single Throw (DPST): One coil is used to operate independent SPST switches at the same time. Useful for switching two loads at the same time.
4. Double Pole, Double Throw (DPDT): This type of relay uses one coil to operate two independent DPDT switches at the same time. This relay can be configured as an H-bridge circuit.

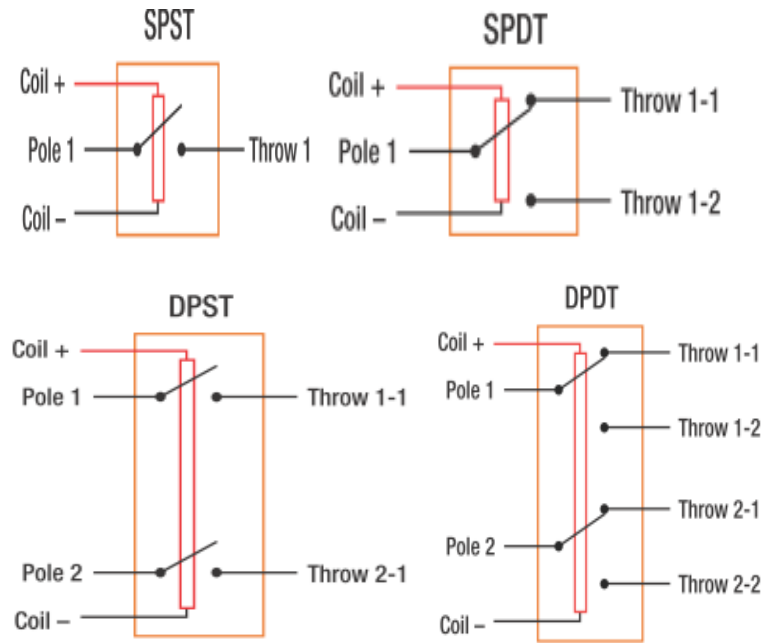


Figure 15: Relay Configurations [12]

CHAPTER 3 METHODOLOGY

3.1 Research Methodology

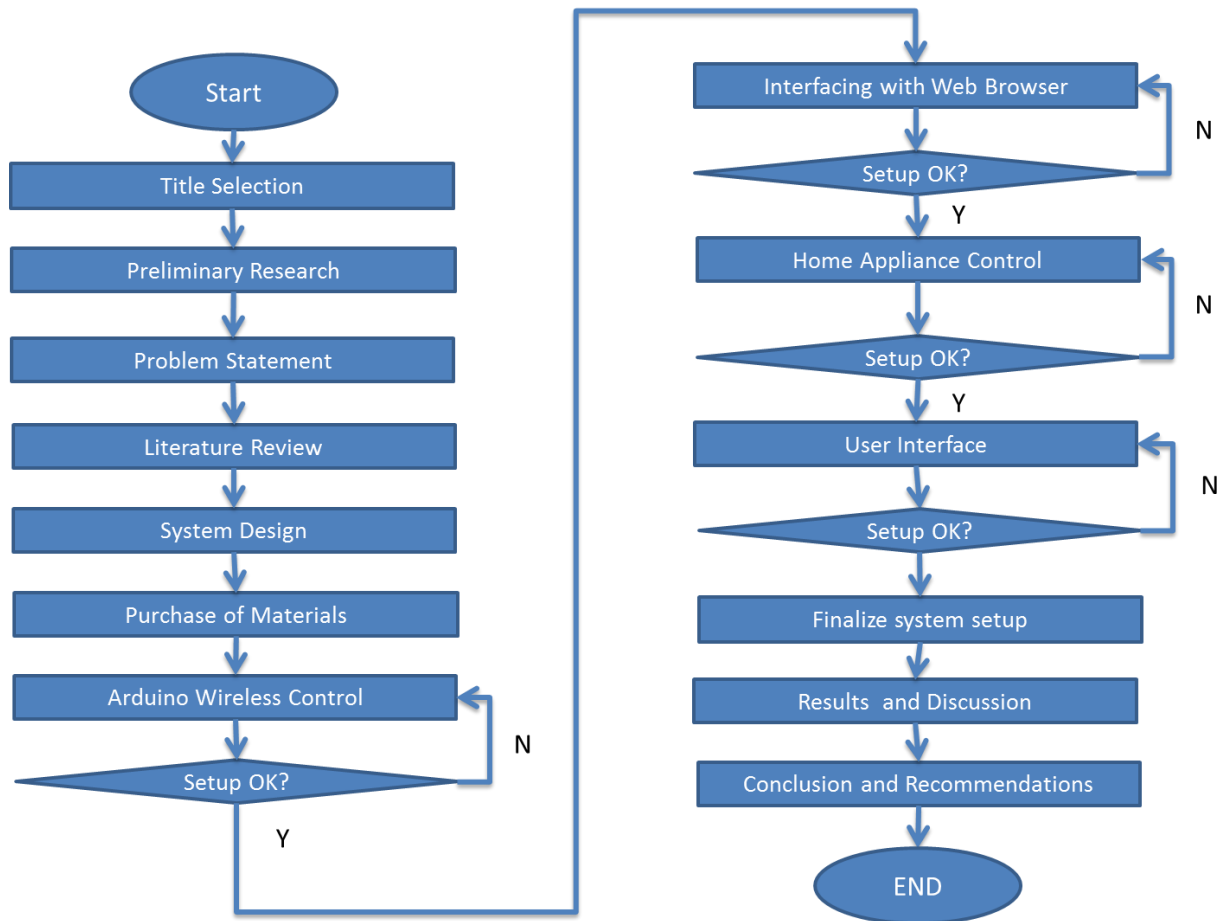


Figure 16: Research Methodology Process Flow Chart

After confirmation of the project title, preliminary research is done to analyze the existing home automation systems to find the problems before recommending a more practical approach. Further research is then made to find the most suitable hardware and software in executing the project. The plan for the system's design is then laid out and materials are purchased.

The Arduino programming language must be familiarized. The next step involves getting the system to work wirelessly utilizing the Wi-Fi shield and wireless control through web browser to control electronic appliances. After the hardware setup is finalized, a simple graphical user interface is created using HTML. Once the system is found to be stable, a more practical setup will be built having the microcontroller to control the actual electronic appliances.

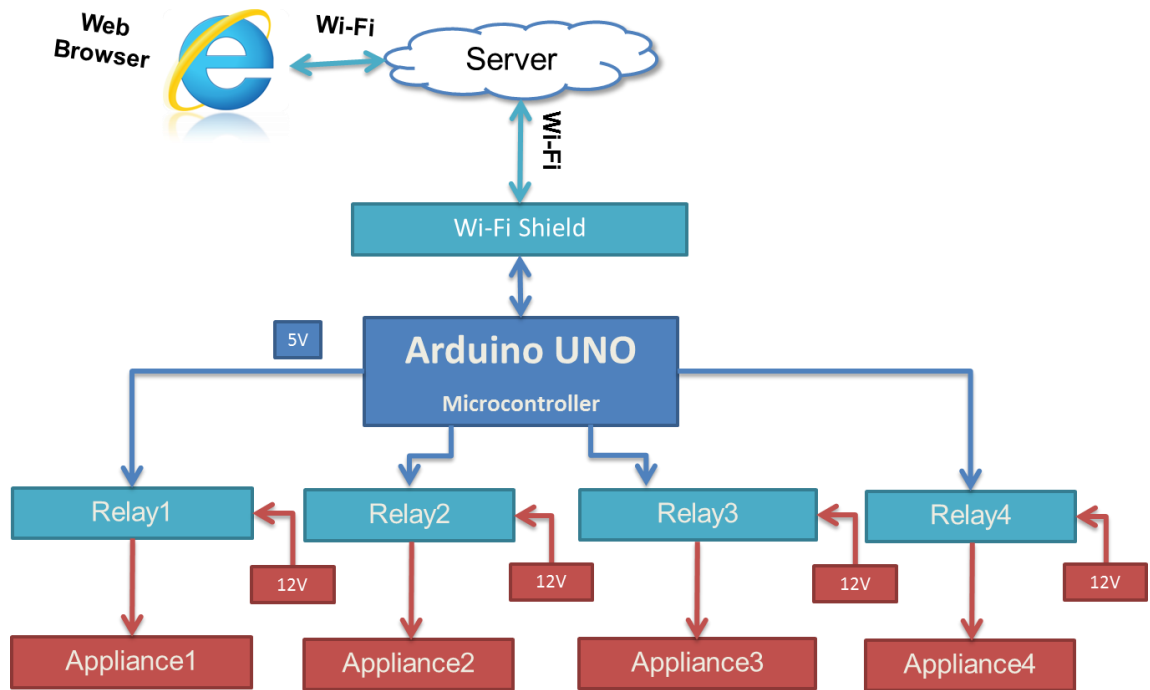


Figure 17: Overall System Design

3.2.2 Final Year Project II Key Milestones:

- Completion of System Setup : Week 4
- Completion of Web Browser Interface Integration : Week 7
- Completion of Analysis of Results : Week 10
- Completion of Discussion : Week 12
- Completion of Conclusion : Week 14

3.3 Tools

In this section, a list of the hardware and software to be used is listed.

3.3.1 Hardware

- Arduino Uno – the microcontroller of choice for this project
- Arduino Wi-Fi Shield
- Solid State Relays
- Resistors
- Maxis 3G Broadband
- Electronic Appliances (Light bulb, 12 Volt Motor)
- 12 Volt Power Supply

3.3.2 Software

- Arduino 1.0.3 (Arduino Software)
- Arduino Development Environment
- Web Browser (e.g. Google chrome, Internet Explorer, Firefox)

CHAPTER 4

RESULTS AND DISCUSSION

4.1 System Setup

For the system setup, Maxis 3G broadband was used as wireless router to connect the client (web browser) and webserver (Arduino). Serial monitor on PC was used to monitor the output of the Arduino.

As the Arduino operates on 5 volts, 5V single pole double throw (SPDT) relays are used to allow control for appliances with higher voltage ratings. In this case, the appliances, consisting of light bulbs and motors, all operate on 12 volts; hence a 12 volt AC to DC adapter is used as external power supply.

Figure 18 and Figure 24 illustrate the circuit diagram and block diagram of the system setup respectively.

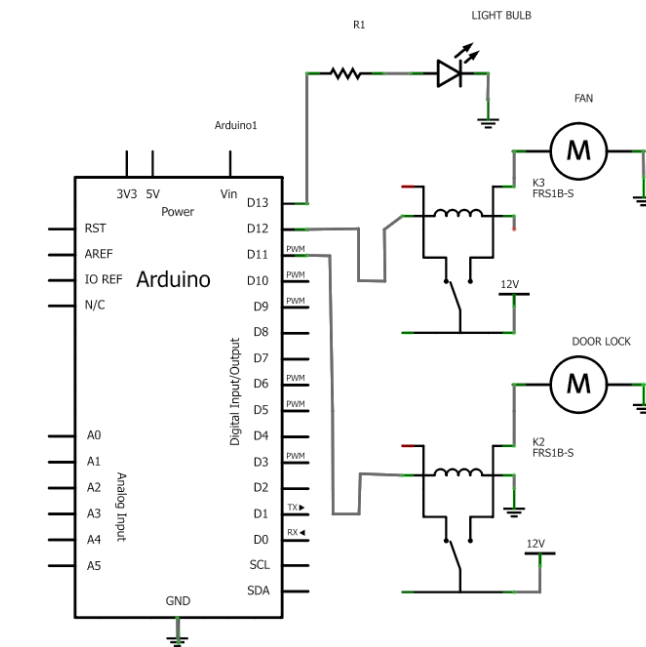


Figure 18: Circuit Diagram for the Home Automation System. The LED represents the lighting control, while the two motors represent ventilation and security control

4.2 System Operation

An overview of the system operation can be seen in Figure 24. Commands are sent to the Arduino via HTTP requests made through the web browser's address bar using the following format:

“IP address”/ “Operation” “Appliance ID”

- *“IP address”* is the IP address of the Arduino which can be obtained from the serial monitor. The Arduino is programmed to broadcast its IP address upon successful connection to the network.
- *“Operation”* can be either an ‘ON’ or ‘OFF’ command, while *“Appliance ID”* is the identification number of the appliance of interest. In this setup, ‘\$’ symbol was chosen for ‘ON’ command while ‘@’ symbol for OFF. Numbers were assigned to appliances to identify which appliance the Arduino should send the command to.

Below is an example of the operation.

Example: Switching on the light bulb

Before sending commands to the Arduino, the Arduino's IP address must first be identified. This can easily be done by simply opening the serial monitor on the PC. As soon as the Arduino connects, its IP address will be printed to the monitor. Figure 19 shows a screen shot of the serial monitor display for this example.

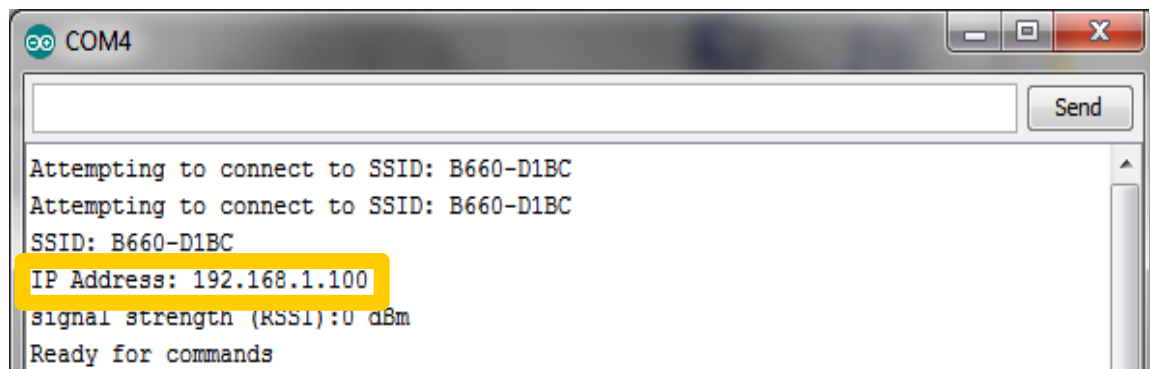


Figure 19: Serial monitor showing the Arduino's activity. The IP address is boxed in orange

In this example, the IP address of the Arduino was found to be “192.168.1.100” and the light bulb was assigned an ID of ‘1’. Hence in this case sending an ON command for the light bulb to the Arduino would be “192.168.1.100/\$1”. Figure 20 shows an example for sending an ON command for the light bulb.



Figure 20: Sending an HTTP request to the Arduino using Firefox web browser. In this case an ON command for the light bulb is being sent to the Arduino

Upon receiving the HTTP request, the Arduino parses the query string for any commands. In this case the command ‘\$1’ has been received.

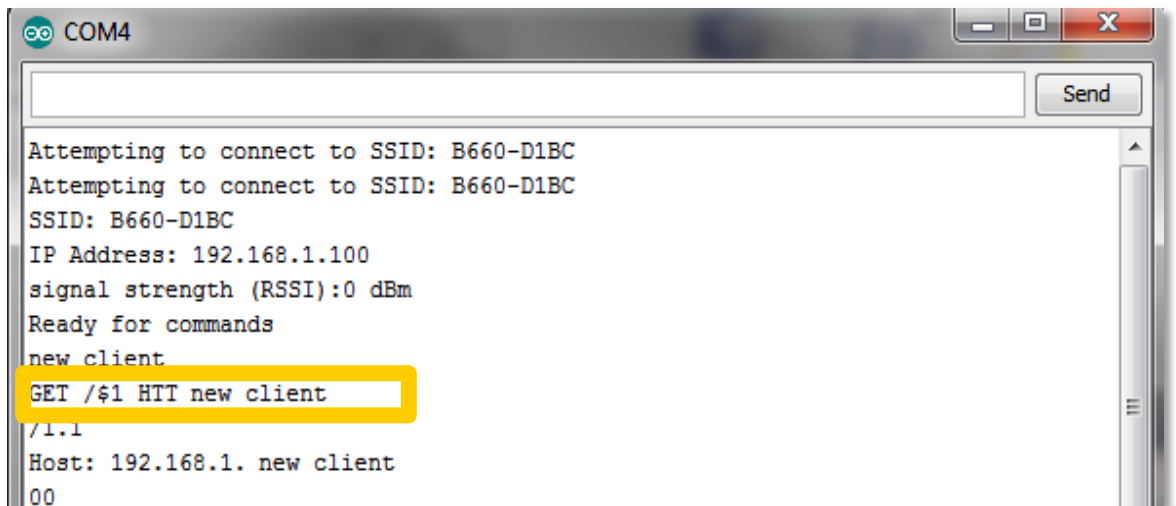


Figure 21: Serial monitor showing the HTTP request received by the Arduino

The Arduino immediately carries out the command received, turning on the light bulb.

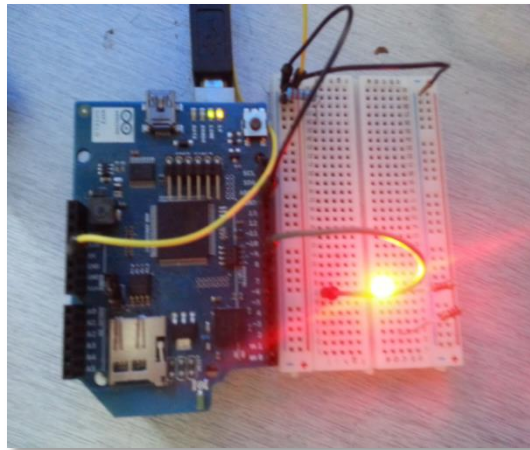


Figure 22: After receiving command light bulb is switched ON

At the same time, the Arduino sends an HTTP response to the web browser notifying the user that the light bulb has been turned ON.

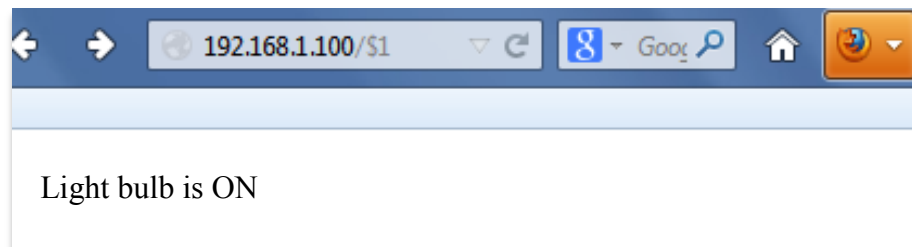


Figure 23: Web browser showing the HTTP response sent by the Arduino

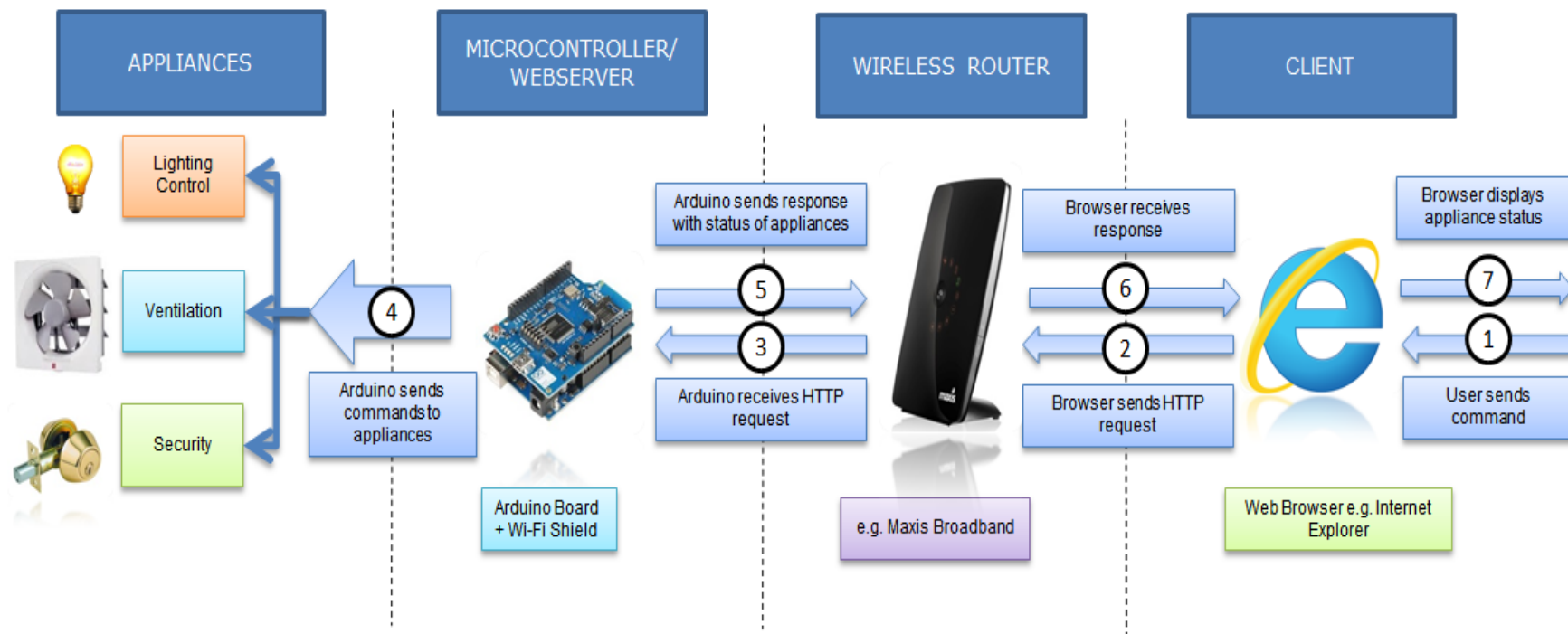


Figure 24: Block diagram representing system setup and operation

4.3 Program

The program consists of two main parts

- Setup and server connection
- Data processing

Each part will be discussed below.

4.3.1 Setup and Server Connection

In the first part of the program involves setup and initialization. The WiFi library, string library, serial library, server IP address and ports are initialized and the Arduino pin modes are set. Afterwards the program continuously tries to login to the wireless network using the specified credentials until login is successful. Upon successful connection, the SSID, WiFi shield IP address and received signal strength are printed to the serial monitor. The program then continues on to connect to the server.

4.3.2 Data processing

The second part of the program involves data processing. The data is first received from the server through an HTTP request, and is read character by character. When an ON or OFF command symbol is identified ('\$' for ON and '@' for OFF), the program continues to read for the next character, which is the appliance ID. For each appliance ID a corresponding port on the Arduino is assigned. If the ON command symbol was identified, the corresponding port is set HIGH activating the relay connected to the appliance, switching on the appliance. The opposite is true for the OFF command, the corresponding port is set LOW deactivating the relay connected to the appliance turning the appliance OFF.

The flow chart in Figure 25 below shows the process flow of the program uploaded to the Arduino UNO.

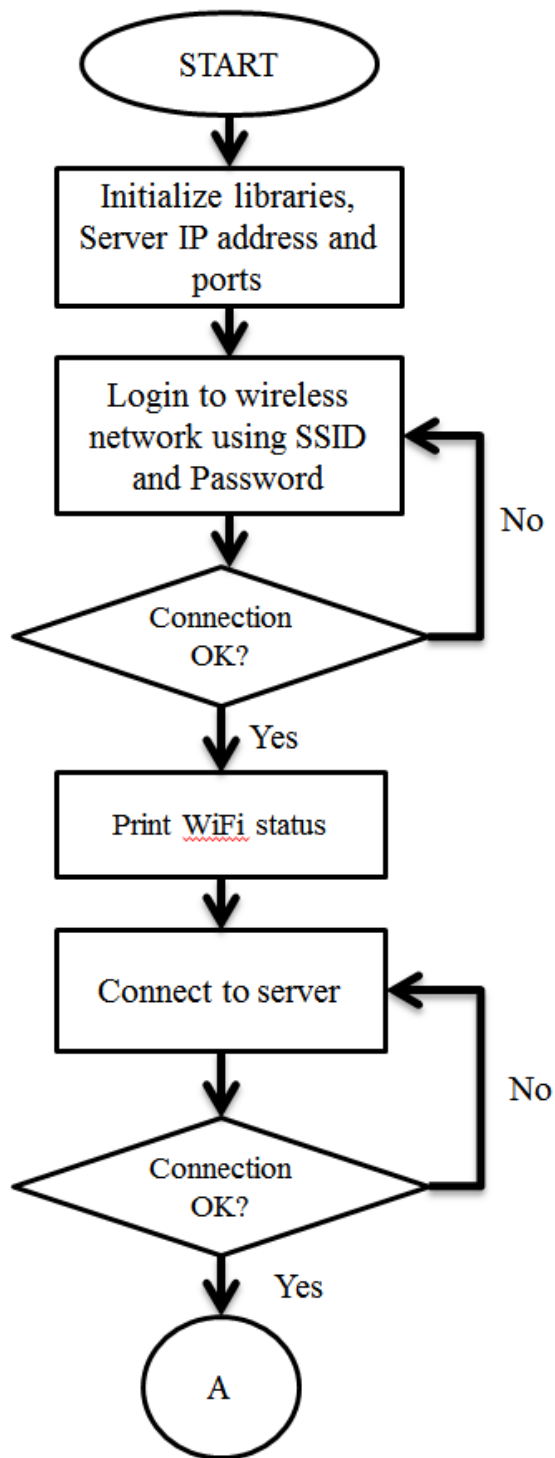


Figure 25: Arduino Program Process Flowchart: Setup and Server Connection

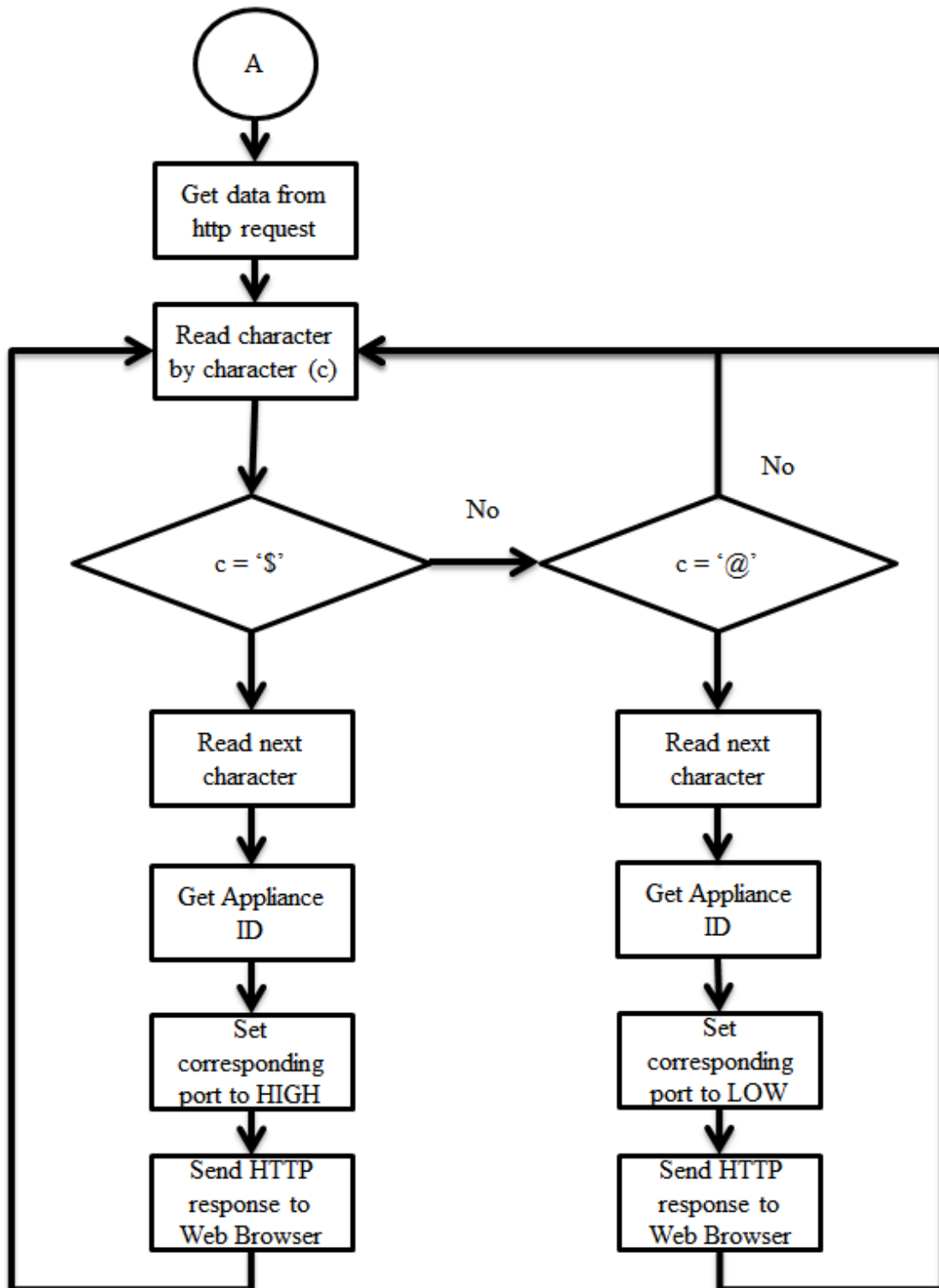


Figure 26: Arduino Program Process Flowchart: Data processing

CHAPTER 5

CONCLUSION AND RECOMMENDATIONS

The home automation system design was successfully implemented. Using web browser on laptop and smartphone, connection was made to Arduino through wireless network, allowing for control of the appliances in the home model.

For future enhancements, sensors such as temperature, light, and motion detection sensors can be introduced into the system to make the home automation system even more intelligent. For example, motion detectors can automatically switch on lights if occupants enter a room, and switch off the lights after occupants have left.

Remote internet access is another useful feature that can be implemented in the home automation system. This will allow the user to control and monitor the home system from anywhere and anytime.

With the completion of this project, it is hoped that an open source, user friendly and affordable home automation system will be possible for everyone.

CHAPTER 6

REFERENCES

- [1] T. Begum, M. S. Hossain, M. B. Uddin and M. S. H. Chowdhury, "Design and Development of Activation and Monitoring of Home Automation System via SMS through Microcontroller," *International Conference on Computers and Devices for Communication*, 2009.
- [2] K. Gill, S.-H. Yang, F. Yao and X. Lu, "A ZigBee-Based Home Automation System," *IEEE Transactions on Consumer Electronics*, vol. 55, no. 2, 2009.
- [3] H. AlShu'eili, G. S. Gupta and S. Mukhopadhyay, "Voice Recognition Based Wireless Home Automation System," *2011 4th International Conference on Mechatronics (ICOM)*, 2011.
- [4] T. Starner, J. Auxier and D. A. M. Gandy, "The Gesture Pendant: A Self-illuminating, Wearable, Infrared Computer Vision System for Home Automation Control and Medical Monitoring," *IEEE*, 2000.
- [5] D. Wheat, *Arduino Internals*, Apress, 2011.
- [6] "Arduino," Arduino, [Online]. Available: <http://arduino.cc>. [Accessed 2 February 2013].
- [7] "code.google.com," Arduino, [Online]. Available: <http://code.google.com/p/arduino/>. [Accessed 19 February 2013].
- [8] S. F. Barrett, *Arduino Microcontroller: Processing for Everyone! Second Edition*, Morgan & Claypool Publishers, 2012.
- [9] T. Igoe, *Making Things Talk 2nd Edition*, Sebastopol: O'REILLY, 2011.
- [10] "Arduino WiFi Shield," Arduino, [Online]. Available: <http://arduino.cc/en/Main/ArduinoWiFiShield>. [Accessed 8 March 2013].
- [11] T. Dean, "Network+ Guide to Networks," in *Network+ Guide to Networks Fifth Edition*, Boston, Course Technology, Cengage Learning, 2010, pp. 364-383.
- [12] J. D. Warren and H. M. Josh Adams, *Arduino Robotics*, New York: Apress, 2011.

APPENDICES

APPENDIX A

ARDUINO PROGRAM CODE

```
/*
  Web Server
*/
#include <SPI.h>
#include <WiFi.h>
#include <String.h>

char ssid[] = "FBD32_MaxisBroadband"; // your network SSID (name)
char pass[] = "9R75E0NQET"; // your network password

int status = WL_IDLE_STATUS;
int state;

String string1 = String(100); //string for storing Light Status
String string2 = String(100); //string for storing Motor Status

WiFiServer server(80);

void setup() {
  //Initialize serial and wait for port to open:
  pinMode(9, OUTPUT);
  pinMode(8, OUTPUT);
  Serial.begin(9600);
  while (!Serial) {
    ; // wait for serial port to connect. Needed for Leonardo only
  }

  // attempt to connect to Wifi network:
  while ( status != WL_CONNECTED) {
    Serial.print("Attempting to connect to SSID: ");
    Serial.println(ssid);
    // Connect to WPA/WPA2 network. Change this line if using open or WEP network:
    status = WiFi.begin(ssid, pass);

    // wait 10 seconds for connection:
    delay(10000);
  }
  server.begin();
  // you're connected now, so print out the status:
  printWifiStatus();
  //digitalWrite(9, LOW);
  Serial.println("Ready for commands");
}

void loop() {
  // listen for incoming clients
  WiFiClient client = server.available();
  if (client) {
    Serial.println("new client");
    client.println("Successfully Connected");

    //client.println("HTTP/1.0 200 OK\r\nContent-Type: text/html\r\n\r\n");
    //client.println("<center><p><h1><a href='\"http://www.hobbyist.co.nz/\">\"www.hobbyist.co.nz/\"</a> presents WebSwitch!
    </h1></p>");
    //client.println("<hr><br><FORM METHOD='\"LINK\"\" ACTION='\"http://192.168.1.101/$1\"\">");
    //client.println("<INPUT TYPE='\"submit\"\" VALUE='\"LightON\"\">");
    //client.println("</FORM>");

    // an http request ends with a blank line
    boolean currentLineIsBlank = true;
    while (client.connected()) {
      if (client.available()) {
```

```

char c = client.read();
Serial.write(c);
switch(c)
{
  case '$':
    c = client.read();
    Serial.write(c);
    switch(c)
    {
      case '1':
        digitalWrite(9, HIGH);
        string1 = "Light : ON";
        client.println("<!DOCTYPE HTML>");
        client.println("<html>");
        client.print("Light is ON");
        client.println("<br />");
        client.println("</html>");
        break;

      case '2':
        digitalWrite(8, HIGH);
        delay(1000);
        string2 = "Motor : ON";
        client.println("<!DOCTYPE HTML>");
        client.println("<html>");
        client.print("Motor is ON");
        client.println("<br />");
        client.println("</html>");
        break;

    }
  break;

  case '@':
    c = client.read();
    Serial.write(c);
    switch(c)
    {
      case '1':
        digitalWrite(9,LOW);
        string1 = "Light : OFF";
        client.println("<!DOCTYPE HTML>");
        client.println("<html>");
        client.print("Light is OFF");
        client.println("<br />");
        client.println("</html>");
        break;

      case '2':
        digitalWrite(8,LOW);
        string1 = "Motor : OFF";
        client.println("<!DOCTYPE HTML>");
        client.println("<html>");
        client.print("Motor is OFF");
        client.println("<br />");
        client.println("</html>");
        break;
    }
  break;
}

// if you've gotten to the end of the line (received a newline
// character) and the line is blank, the http request has ended,
// so you can send a reply

if (c == '\n' && currentLineIsBlank) {
  Serial.println("Reached end of http request");
  // send a standard http response header
  client.println("HTTP/1.1 200 OK");
  client.println("Content-Type: text/html");
  client.println("Connection: close");
  client.println();
  client.println("<!DOCTYPE HTML>");
  client.println("<html>");
}

```

```

    // add a meta refresh tag, so the browser pulls again every 5 seconds:
    // client.println("<meta http-equiv=\"refresh\" content=\"5\">");
    // output the value of each analog input pin
    client.println(string1);
    client.println(string2);
    client.println("<br />");
    client.println("</html>");
    break;
}

if (c == '\n') {
    // you're starting a new line
    currentLineIsBlank = true;
}
else if (c != '\r') {
    // you've gotten a character on the current line
    currentLineIsBlank = false;
}
}
// give the web browser time to receive the data
delay(1000);
// close the connection:
client.stop();
Serial.println("client disconnected");
}
}

void printWifiStatus() {
    // print the SSID of the network you're attached to:
    Serial.print("SSID: ");
    Serial.println(WiFi.SSID());

    // print your WiFi shield's IP address:
    IPAddress ip = WiFi.localIP();
    Serial.print("IP Address: ");
    Serial.println(ip);

    // print the received signal strength:
    long rssi = WiFi.RSSI();
    Serial.print("signal strength (RSSI):");
    Serial.print(rssi);
    Serial.println(" dBm");
}

```