Kurdistan Region

University of Salahaddin-Erbil

College of Engineering

Electrical Engineering Department

Home Automation System Using Arduino

A project Submitted to the Electrical Engineering Department University of Salahaddin –Erbil in the Partial Fulfillment of the Requirement for the Degree of Bachelor of Science in Electrical Engineering

Prepared by:

Houshang Essmat Eziz

Supervisor:

Mr. Ahmad Khalid Ahmad

June-2020

Supervisor's Certification

I certify that the Engineering project titled "Home Automation System Using Arduino" was done under my supervision at the Electrical Engineering Department, College of Engineering –Salahaddin University-Erbil. In the partial fulfillment of the requirement of the degree of Bachelor of Science in Electrical Engineering.

Supervisor Signature: [illegible] ----------------------------

Name: Mr. Ahmad Khalid Ahmad

Date: 10 / 06/2020

DEDICATION

I would like to thank everyone who has helped in completion of this thesis work, for their advice, suggestion and help. I cordially thank my supervisor Mr. Ahmad Khalid Ahmad. This thesis would not have been possible without their continuous support. Finally, I thank my beloved parents for their never-ending support, motivation and belief in me.

ABSTRACT

This project proposes a new way to control home appliances using Arduino board from internet. This makes it very simple allows everyone to be able to use it because all you have to do is access the Arduino through a web browser on any device attached to a network.

LIST OF CONTENT

LIST OF FIGURES

CHAPTER 1: INTRODUCTION

1.1 Background

As technology advances in our daily lives, it has become far more important to everyone. This means that our reliance upon technology has increased at a dramatic rate. One of the areas which is affected by this is home automation. Each day we expect our appliances and devices to perform functions automatically and without manual interaction. This ranges from small tasks like the washing machine turning off when it has finished, all the way through safety critical systems having a procedure in place should a catastrophic event happen. However, many appliances simply cannot automate themselves at this current time, whether this is due to cost, lack of demand or simply manufacturer oversight.

By using off the shelf hardware such as an Arduino board, we can enable existing devices (which are not capable of automation by themselves) to be controlled remotely. However, this hardware is capable of powering more than just basic appliances and digital devices. Lighting, curtains, and other everyday items can be connected to automatically perform user specified operations, allowing a huge array of everyday items to perform specific functions using the home automation system.

1.2 Home Automation:

Home automation (also known as domotics) refers to the automatic and electronic control of household features,activity, and appliances.Various control systems are utilized in this residential extension of building automation .some components of an automated home may include the centralized control of security locks on doors and gates,appliances, windows and lighting.

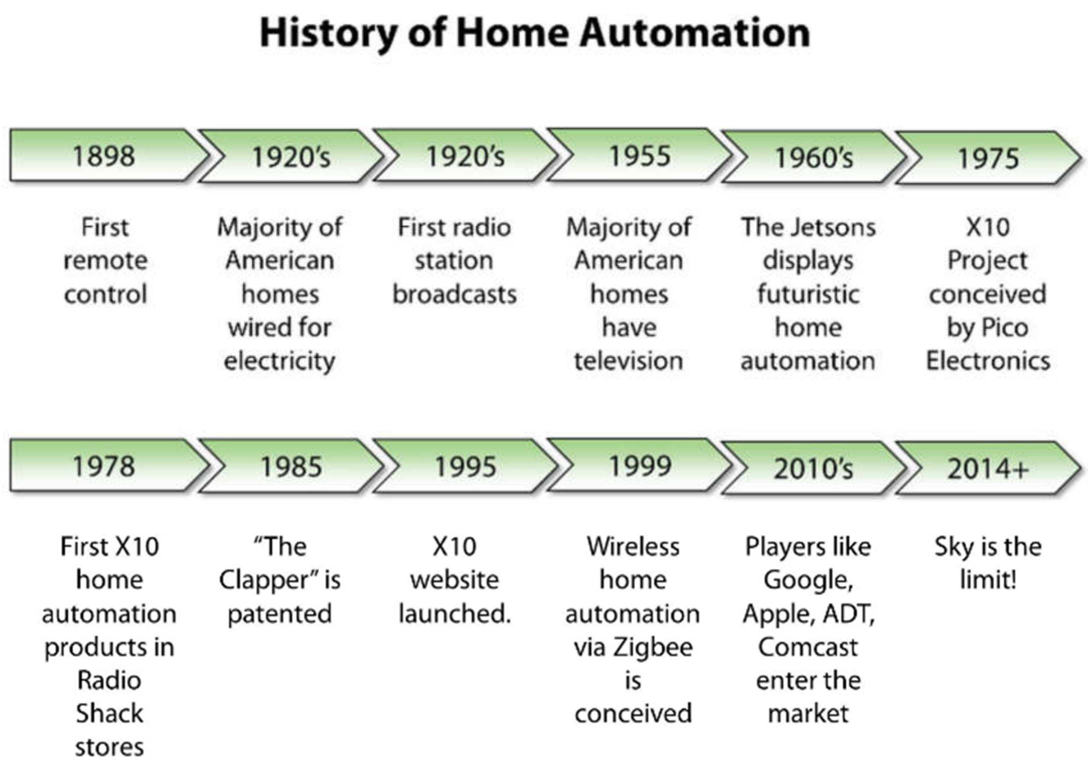

History of Home Automation
1898
First remote control
1920's
Majority of American homes wired for electricity
1920's
First radio station broadcasts
1955
Majority of American homes have television
1960's
The Jetsons displays futuristic home automation
1975
X10 Project conceived by Pico Electronics
1978
First X10 home automation products in Radio Shack stores
1985
"The Clapper" is patented
1995
X10 website launched.
1999
Wireless home automation via Zigbee is conceived
2010's
Players like Google, Apple, ADT, Comcast enter the market
2014+
Sky is the limit!

Figure 1-1 home automation history

1.3 Problem Statement:

Nowadays people are very busy with work and are always in a hurry to get out of the house. This causes many of us to either forget to turn off an appliance or no longer have control of the appliances once we are out of the house. But as you all know these days' people have access to the internet everywhere they go, therefore we can develop a way for people to control their home appliances remotely even when they are on the go.

1.4 Aims and Objectives:

There are many home automations in Kurdistan, so my objective is to develop a way that provide much better, efficient and easier to controlling appliances wirelessly by using Arduino through web interface.

1.5 Arduino

Arduino is an open-source prototyping platform based on easy-to-use hardware and software. Arduino, boards are able to read inputs -light on a sensor, a finger on a button, or a Twitter message -and turn it into an output -activating a motor, turning on an LED, publishing something online. You can tell your board what to do by sending a set of instructions to the microcontroller on the board. To do so you use the Arduino programming language (based on Wiring), and the Arduino Software (IDE), based on Processing, the following figure is an image of Arduino board.

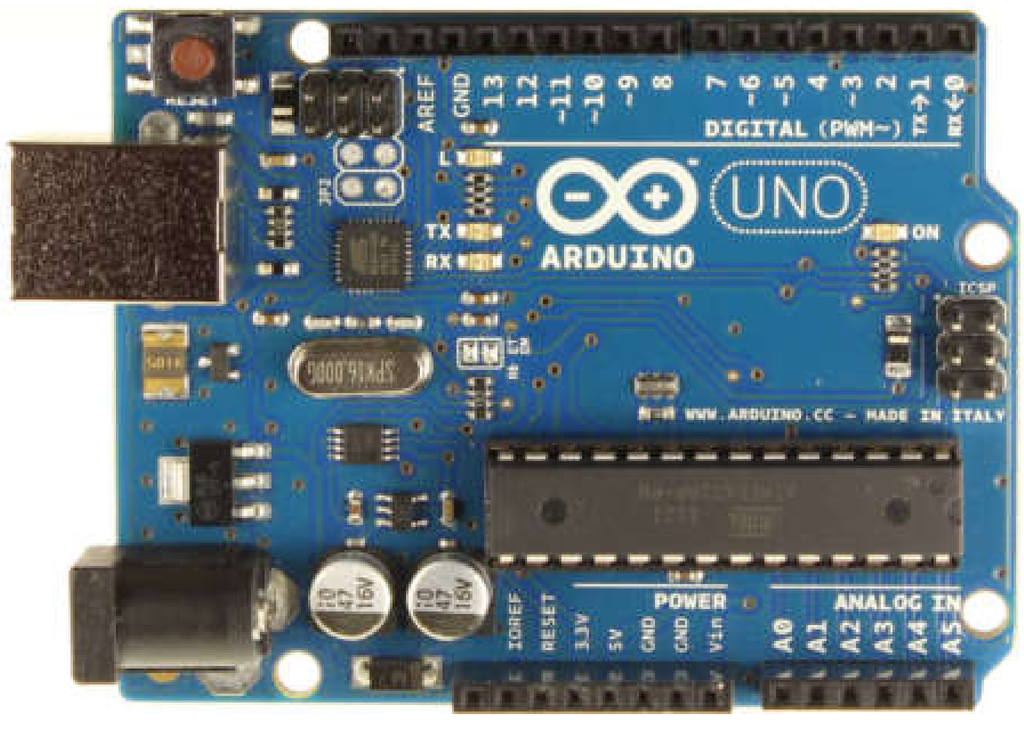


Figure 1-2: Arduino Board

1.5.1 Why Arduino?

Arduino has been used in thousands of different projects and applications. The Arduino software is easy-to-use for beginners, yet flexible enough for advanced users. It runs on Mac, Windows, and Linux. Teachers and students use it to build low cost scientific instruments, to prove chemistry and physics principles and to get started with programming and robotics.

1.5.2 Advantages of Arduino:

Inexpensive: Arduino boards are relatively inexpensive compared to other microcontroller platforms. The least expensive version of the Arduino module can be assembled by hand, and even the pre-assembled Arduino modules cost less than $50.

Cross-platform: The Arduino Software (IDE) runs on Windows, Macintosh OSX, and Linux operating systems. Most microcontroller systems are limited to Windows.

Simple, clear programming environment: The Arduino Software (IDE) iseasy-to-use for beginners, yet flexible enough for advanced users to take advantage of as well. For teachers, it's conveniently based on the Processing programming environment, so students learning to program in that environment will be familiar with how the Arduino IDE works.

Open source and extensible software: The Arduino software is published as open source tools, available for extension by experienced programmers. The language can be expanded through C++ libraries, and people wanting to understand the technical details can make the leap from Arduino to the AVR C programming language on which it's based. Similarly, you can add AVR-C code directly into your Arduino programs if you want to.

Open source and extensible hardware: The plans of the Arduino boards are published under a Creative Commons license, so experienced circuit designers can make their own version of the module, extending it and improving it. Even relatively inexperienced users can build the breadboard version of the module in order to understand how it works and save money.

1.6 Organization of the Report:

Chapter 1: In this chapter I will give a brief introduction to my project and clarify what Arduino board is and why I selected it for my project

Chapter 2: Here I will mention the Arduino hardware specifications that will be needed for the completion of my project.

Chapter 3: This chapter will include details about the Arduino software Programming language.

Chapter 4: This chapter will contain information about other developments that are related to my project.

Chapter 5: This chapter will include the conclusion of my report.

CHAPTER 2:

2.1 Arduino Hardware Specifications

Arduino is a software company, project, and user community that designs and manufactures computer open-source hardware, open-source software, and microcontroller-based kits for building digital devices and interactive objects that can sense and control physical devices. The project is based on microcontroller board designs, produced by several vendors, using various microcontrollers. These systems provide sets of digital and analog I/O pins that can interface to various expansion boards (termed shields) and other circuits. The boards feature serial communication interfaces, including Universal Serial Bus (USB) on some models, for loading programs from personal computers. For programming the microcontrollers, the Arduino project provides an integrated development environment(IDE) based on a programming language named Processing, which also supports the languages C and C++.The first Arduino was introduced in 2005, aiming to provide a low cost, easy way for novices and professionals to create devices that interact with their environment using sensors and actuators. Common examples of such devices intended for beginner hobbyists include simple robots, thermostats, and motion detectors.

Arduino boards are available commercially in preassembled form, or as do-it-yourself kits. The hardware design specifications are openly available, allowing the Arduino boards to be produced by anyone. Adafruit Industries estimated inmid-2011 that over 300,000 official Arduino had been commercially produced ,and in 2013 that 700,000 official boards were in users' hands.

2.2 Arduino Hardware

2.2.1 Arduino Board

Arduino board historically consists of an Atmel 8-, 16-or 32-bit AVR microcontroller (although since 2015 other makers' microcontrollers have been used) with complementary components that facilitate Programming and incorporation into other circuits. An important aspect of the Arduino is its standard connectors, which let users connect the CPU board to a variety of interchangeable add-on modules termed shields. Some shields communicate with the Arduino board directly over various pins, but many shields are individually addressable via anI^2Cserial bus—so many shields can be stacked and used in parallel. Before 2015, Official Arduino's had used the Atmelmega AVR series of chips, specifically the ATmega8, ATmega168, ATmega328, ATmega1280, andATmega2560. In 2015, units by other producers were added. A handful of other processors have also been us by Arduino compatible devices. Most boards include a 5 V linear regulator and a 16MHz crystal oscillator (or ceramic resonator in some variants), although some designs such as the Lily Pad run at 8MHz and dispense with the onboard voltage regulator due to specific form-factor restrictions. An Arduino's microcontroller is also pre-programmed with a bootloader that simplifies uploading of programs to the on-chip flash memory, compared with other devices that typically need an external programmer. This makes using an Arduino more straightforward by allowing the use of an ordinary computer as the programmer. Currently, Opti bootloader is the default boot loader installed on Arduino UNO.

An early Arduino board with an RS-232 serial communication interface (upper left) and an Atmel ATmega8 microcontroller chip (black, lower right); the 14 digital I/O pins are located at the top and the six analog input pins at the lower right.

At a conceptual level, when using the Arduino integrated development environment, all boards are programmed over a serial connection. Its Implementation varies with the hardware version. Some serial Arduino Boards contain a level shifter circuit to convert betweenRS-232Logic levels and transistor–transistor logic (TTL) level signals. Current Arduino boards are programmed via Universal Serial Bus (USB), implemented using USB-to-serial adapter chips such as the FTDI FT232. Some boards, such as later-model Uno boards, substitute the FTDI chip with a separate AVR chip containing USB-to-serial firmware, which is reprogrammable via its own ICSP header. Other variants, such as the Arduino Mini and the unofficial Boarduino, use a detachable USB-to-serial adapter board or cable, Bluetooth or other methods, when used with traditional microcontroller tools instead of the Arduino IDE, standard AVR in-system programming (ISP) programming is used. An official Arduino Uno Revision 2 with descriptions of the I/O Locations m he Arduino board exposes most of the microcontroller's I/O pins for use by other circuits. The decimal, Duemilanove [b], and current Uno[c] provide 14 digital I/O pins, six of which can produce pulse-width modulated signals, and six analog inputs, which can also be used as six digital I/O pins.
These pins are on the top of the board, via female 0.1-inch (2.54 mm) headers. Several plug-in application shields are also commercially available. The Arduino Nano and Arduino-compatible Bare Bones.
Board and Boarduinoboards may provide male header pins on the underside of the board that can plug into solder less breadboards.

Many Arduino-compatible and Arduino-derived boards exist. Some are functionally equivalent to an Arduino and can be used interchangeably. Many enhance the basic Arduino by adding output drivers, often for use in school-level education, to simplify making buggies and small robots. Others are electrically equivalent but change the form factor, sometimes retaining compatibility with shields, sometimes not. Some variants use different processors, of varying compatibility.

2.2.2 Official Boards

The original Arduino hardware was produced by the Italian company Smart Projects .Some Arduino-branded boards have been designed by the American companies Spark Fun Electronics and Adafruit Industries. as of 2016, 17 versions of the Arduino hardware had been commercially produced.

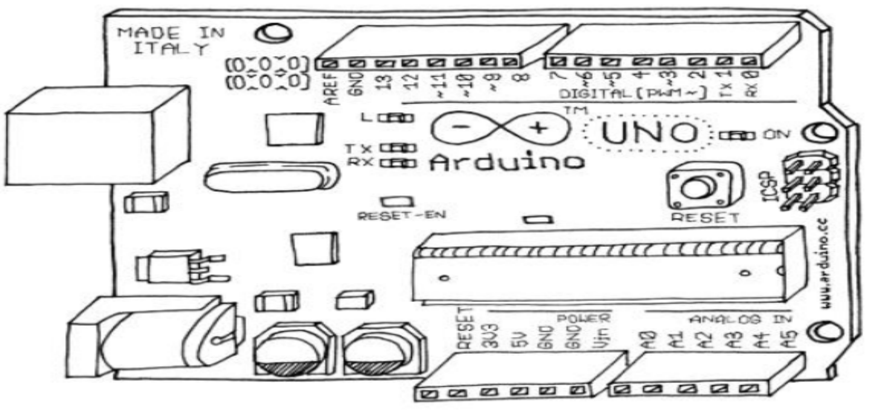


Figure 2-1: Arduino Uno

IO Pins of Arduino

Here is an explanation of what every pins of the board do:

14 Digital IO pins (pins 0–13): These can be inputs or outputs, which is specified by the sketch you create in the IDE.

6 Analogue In pins (pins 0–5): These dedicated analogue input pins take analogue values (i.e., voltage readings from a sensor) and convert them into a number between 0 and1023.

6 Analogue Out pins (pins 3, 5, 6, 9, 10, and 11): These are actually six of the digital pins that can be reprogrammed for analogue output using the sketch you create in the IDE.

The board can be powered from your computer's USB port, most USB chargers, or an AC adapter (9 volts recommended, 2.1mm barrel tip,center positive). If there is no power supply plugged into the power socket, the power will come from the USB board, but as soon as you plug a power supply, the board will automatically use.

2.2.3 Cuhead WIFI Shield V2.0:

This document is to introduce CuheadWiFi Shield V2.0 (hereafter we callCuheadV2.0). Cuhead V2.0 use low consumption MRF24WB0MA embedded Wi-Fi transceiver Module match 2.4 GHz IEEE 802.11b™ RF Standard.Cuhead V2.0 adopt standard Arduino laminated design. It is designed to plug on ArduinoDiecimila/Duemilanove/Uno etc.

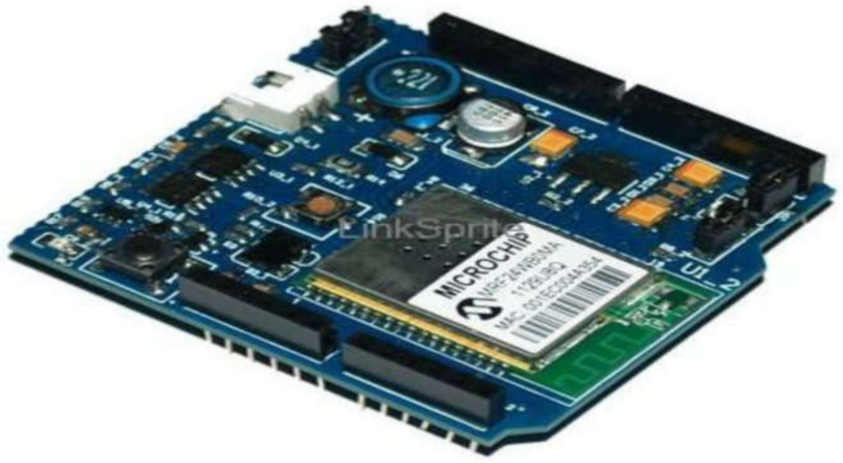


Figure 2.2 Cuhead Wi-Fi Shield

CuheadWiFi Shield connects your Arduino to the internet wirelessly.

Wifishield is stacked on the Arduino board. This Wi-Fi shield can connect to wireless networks which operate according to the 802.11b and 802.11g specification. Also, this Wi-Fi shield can be used store the web that are connected to the Arduino board. Cuhead V2.0 has charging and discharging function, the charging circuit is used to tell voltage of the battery. We can connect the positive and negative of the battery to BAT, if the battery is full, then Cuhead V2.0 wont charge battery; if it is not, the external battery will be charged. So, you know it is Arduino giving power to Cuhead V2.0 or the opposite way, and change the charging circuit to connect/disconnect based on that. When Cuhead V2.0 connected to Arduino, there are two working status:

1. Connect Arduino with USB/Adaptor, Arduino gives power to Cuhead V2.0, the charging circuit on Cuhead V2.0 will disconnect ;

2. No external power for Arduino, then the charging circuit of Cuhead V2.0 is working, and the battery will power the boards. Normally we give power to Arduino directly, thus the charging circuit is inactive.

CHAPTER 3: ARDUINO PROGRAMMING LANGUAGE

3.1 Arduino Software (Introduction)

Arduino programs may be written in any programming language with
a compiler that produces binary machine code. Atmel provides a development environment for their microcontrollers, AVR Studio and the newer Atmel Studio.
The Arduino project provides the Arduino integrated development Environment (IDE), which is a cross-platform application written in the programming language Java. It originated from the IDE for the languages Processing and Wiring. It is designed to introduce programming to artists and other newcomers unfamiliar with software development. It includes a code editor with features such as syntax highlighting, brace matching, and automatic indentation, and provides simple one-click mechanism to compile and load programs to an Arduino board. A program written with the IDE for Arduino is called a "sketch. The Arduino IDE supports the languages C and C++ using special rules to organize code. The Arduino IDE supplies a software library called Wiring from the Wiring project, which provides many common input and output procedures. A typical Arduino C/C++ sketch consist of two functions that are compiled and linked with a program stub main () into an executable cyclic executive program:

setup (): a function that runs once at the start of a program and that can initialize settings.

loop (): a function called repeatedly until the board powers off.

After compiling and linking with the GNU tool chain, also included with the IDE distribution, the Arduino IDE employs the programavrdude to convert the executable code into a text file in hexadecimal coding that is loaded into the Arduino board by a loader program in the board's firmware

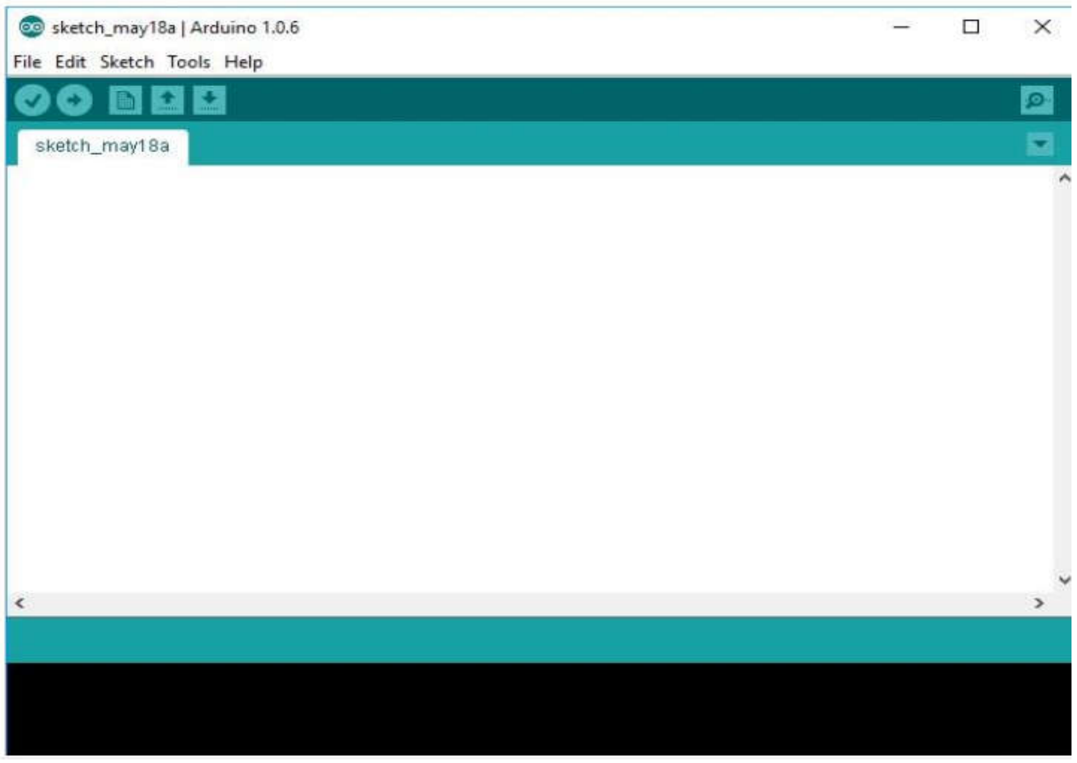
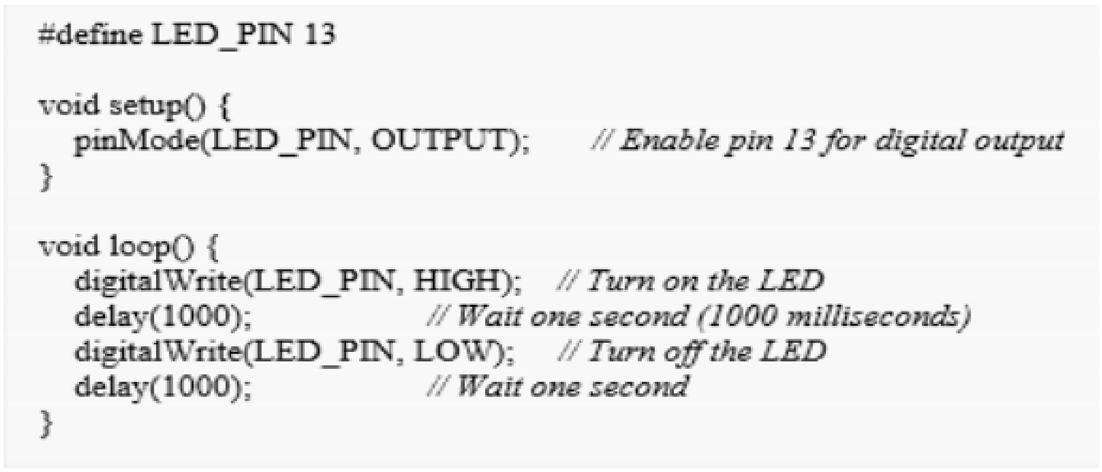


Figure 3-1 programming tool

3.1.1 Sample program

A typical program for a beginning Arduino programmer blinks a light-emitting diode (LED) on and off. This program is usually loaded in the Arduino board by the manufacturer. In the Arduino environment, a user might write such a program as shown:



```
#define LED_PIN 13

void setup() {
   pinMode(LED_PIN, OUTPUT);      // Enable pin 13 for digital output
}

void loop() {
   digitalWrite(LED_PIN, HIGH);   // Turn on the LED
   delay(1000);                   // Wait one second (1000 milliseconds)
   digitalWrite(LED_PIN, LOW);    // Turn off the LED
   delay(1000);                   // Wait one second
}
```

The basic structure of the Arduino programming language is fairly simple andruns in at least two parts. These two required parts, or functions, enclose blocksof statements.

```
void setup ()

{

statements;

}
void loop ()
{
Statement;

}
```

Where setup () is the preparation, loop () is the execution. Both functions are required for the program to work.

The setup function should follow the declaration of any variables at the very beginning of the program. It is the first function to run in the program, is run only once, and is used to set pin Mode or initialize serial communication.

The loop function follows next and includes the code to be executed continuously -reading inputs, triggering outputs, etc. This function is the core of all Arduino programs and does the bulk of the work.

Setup ()

The setup () function is called once when your program starts. Use it to initialize pin modes, or begin serial. It must be included in a program even if there are nostatements to run.

```
void setup()
{
pinMode(pin, OUTPUT); // sets the 'pin' as output
}
```

Loop ()

After calling the setup () function, the loop() function does precisely what its name suggests, and loops consecutively, allowing the program to change,respond, and control the Arduino board.

```
Void loop ()
{
digitalWrite(pin, HIGH); // turns 'pin' on
delay(lOOO); // pauses for one second
digitalWrite(pin, LOW); // turns 'pin' off
delay(lOOO); // pauses for one second
}
```

After calling the setup() function, the loop() function does precisely
what itsname suggests, and loops consecutively, allowing the program to
change,respond, and control the Arduino board.

```
void loop()
{
digitalWrite(pin, HIGH); // turns 'pin' on
delay(lOOO); // pauses for one second
digitalWrite(pin, LOW); // turns 'pin' off
delay(lOOO); // pauses for one second
}
```

3.1.2 Functions

A function is a block of code that has a name and a block of Statements that are executed when the function is called. The functions void setup () and void loop()have already been discussed and other built-in functions will be discussed later. Custom functions can be written to perform repetitive tasks and reduce clutter in
a program. Functions are declared by first declaring the function type. This is the type of value to be returned by the function such as 'int' for an integer type function. If no value is to be returned the function type would be void. After type,declare the name given to the function and in parenthesis any parameters being passed to the function.

```
typefunctionName(parameters)
{
statements;
}
```

The following integer type function delayVal() is used to set a delay value in a program by reading the value of a potentiometer. It first declares a local variable , sets v to the value of the potentiometer which gives a number between 0-1023, then divides that value by 4 for a final value between 0-255, and finally returns that value back to the main program.

```
intdelayVal()
{
int v;//create temporary variable
v = analogRead(pot)//read potentiometer value
v/= 4;//converts 0-1023 to 0-255
return v;//return final value
}
```

Pins configured as OUTPUT are said to be in a low-impedance state and can provide 40 mA (milliamps) of current to other devices/circuits. This is enough current to brightly light up an LED (don't forget the series resistor), but not enough current to run most relays, solenoids, or motors. Short circuits on Arduino pins and excessive current can damage or destroy the output pin, or damage the entire Atmega chip. It is often a good idea to connect an OUTPUT pin to an external device in series with a 470Q or 1KQ resistor.

digitalRead(pin)

Reads the value from a specified digital pin with the result either HIGH or LOW.

The pin can be specified as either a variable or constant (0-13).value = digitalRead(Pin); // sets 'value' equal to
// the input pin

digitalWrite(pin, value)

Outputs either logic level HIGH or LOW at (turns on or off) a specified digital pin.The pin can be specified as either a variable or constant (0-13).

digitalWrite(pin, HIGH); // sets 'pin' to high analogRead(pin)Reads the value from a specified analog pin with a10-bit resolution. This function only works on the analog in pins (0-5). The resulting integer values range from 0 to 1023. value = analog Read(pin); // sets 'value' equal to 'pin'

Note: Analog pins unlike digital ones, do not need to be first declared

as INPUT or OUTPUT.

analogWrite(pin, value)
Writes a pseudo-analog value using hardware enabled pulse width modulation(PWM) to an output pin marked PWM. On newer Arduino with the ATmega168chip, this function works on pins 3, 5, 6, 9, 10, and 11.

Older Arduinos with anATmega8 only support pins 9, 10, and 11. The value can be specified as avariable or constant with a value from 0
-255.analog Write(pin, value); // writes 'value' to analog 'pin'
A value of 0 generates a steady 0 volts output at the specified pin; a value of255 generates a steady 5 volts output at the specified pin. For values in between 0 and 255, the pin rapidly alternates between 0 and 5 volts -the higher the value, the more often the pin is HIGH (5 volts). For example, a value of 64will be 0 volts three-quarters of the time, and 5 volts one quarter of the time; a value of 128 will be at 0 half the time and 255 half the time; and a value of 192will be 0 volts one quarter of the time and 5 volts three-quarters of the time. Because this is a hardware function, the pin will generate a steady wave after a call to analog Write in the background until the next call to analogWrite (or a call to digital Read or digitalWrite on the same pin)

Note:Analog pins unlike digital ones, do not need to be first declared as INPUTnor OUTPUT

CHAPTER 4: RELATED WORK

In this chapter I will explain how I implemented my web interface starting with the hardware components then description of the code used.
The project consists of the following hardware components:
(Arduino UNO Board, CuHeadWiFi Shield, Relay, Bread Board, Resistors, Bulb, LED, Potentiometer)

The Arduino board is used to download the main code which is called Sketch software. This can be done through USB cable connected to a computer running IDE. The communication is a serial communication through the USB cable. Wi-Fi Shield is stacked on the Arduino board. This Wi-Fi Shield can connect to wireless networks which operate according to the 802.11b and 802.11g specifications. Also, this Wi-Fi shield can be used to store the web page that will get requested whenever there is a request to control the LEDs that are connected to the Arduino Board. The LEDs green, yellow, and red are connect to pin numbers 7, 6, and 5, respectively. The green light simulates the water pump that we assumed to fill the tank with water. The two other LEDs, yellow and red, are used to represent the two switch that can be used to control any home appliances, such as TV, Funs, Home Light, or garage door opener. The voltage that we take from the Arduino board is very low and it is a DC voltage, therefore we can use relays instead of LEDs to interface the low DC voltage with normal AC voltage that can operate our home appliances that we need to control. The following picture show the relay that can be connected to Arduino to control a table lamp. To simulate the level of the tank that we need to measure we used the Potentiometer by changing this potentiometer we are simulating the level of the water in the tank. We aimed in this project to turn the water pump automatically whenever the water level is reached the 10% and we wanted to turn the water pump off whenever the water reach 90%. You can also turn the water pump on and off whenever we wanted manually through the web interface. The sketch of the project is written in the Arduino IDE. And it is based on the Simple Server example code with a simple addition to read an analog input and include the value in the web page sent back to the browser. The sketch starts by including the WiShield library. Because we're using it in APP_WISERVER mode we include the WiServer.h The sketch then defines a couple of constant to make the code further down a bit more readable.

```
#include <WiServer.h>

#define WIRELESS_MODE_INFRA 1

#define WIRELESS_MODE_ADHOC 2
```

The sketch needs to know certain configuration values to connect to your WiFi network. These are set using a series of arrays and need to be changed to suit the requirements. The basic network settings are the IP address of your Arduino (which must be unique on your network), the IP address of the router, and the subnet mask for your network. Note that most of the time we have an IP address that is represented in "dotted-quad" format, but in this case each quad is stored as a different element in an array so they have to be separated by commas instead of periods

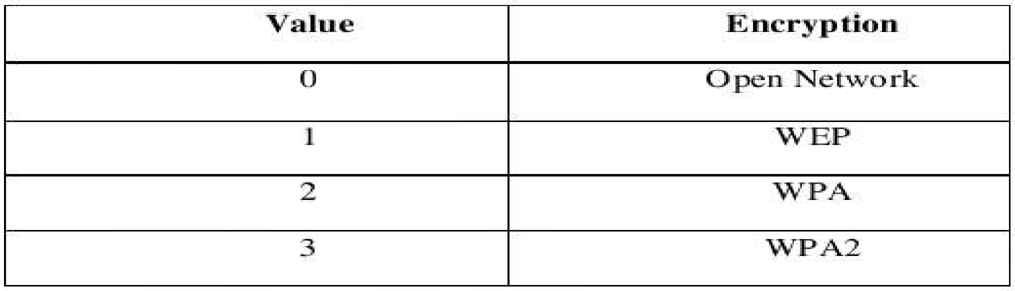
```
unsigned char local_ip[] = {10,0,1,200};

unsigned char gateway_ip[] = {10,0,1,1};
unsigned char subnet_mask[] = {255,255,255,0};
```

The wireless-specific settings start with the SSID (service set identifier) of our access point. This is the Wi-Fi network name that we see on
our computer when selecting a network. Maximum length for the SSID is 32 characters.

```
constprog_charssid[] PROGMEM = {"YourSSID"};
```

We then need to specify the security type. The supported settings are shown in the following Table.



Value	Encryption
0	Open Network
1	WEP
2	WPA
3	WPA2

Figure 4-1 Wi-Fi Server Network Security Mode

On our project we connected to a WPA2-encrypted network, so we set

it to 3.

```
unsigned char security_type = 3;        // 0 - open; 1 - WEP; 2 - WPA; 3 - WPA2
// WPA/WPA2 passphrase
```

Ifyou use WPA or WPA2 you also need to supply the passphrase to join the network. The value can be up to 64 characters long.

```
constprog_charsecurity_passphrase[] PROGMEM = {"smart123"};    // max 64
characters
```

If you are using WEP you need to define the 128-bit WEP key for your network. WEP supports multiple keys and so does the WiShield, so youcan configure them by entering the appropriate hex values into the program.

```
prog_ucharwep_keys[] PROGMEM = { 0x01, 0x02, 0x03, 0x04, 0x05, 0x06,
0x07, 0x08, 0x09, 0x0a, 0x0b, 0x0c, 0x0d,   // Key 0
                              0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00,  // Key 1
                              0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00,  // Key 2
                              0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00   // Key 3
                            };
```

WiFi supports two basic modes: infrastructure and ad-hoc. The most common is infrastructure with each mobile device connecting to a central access point, but it's also possible to run in ad-hoc mode where devices connect directly to their peers. We connected our Arduino to an access point so we set it to WIRELESS_MODE_INFRA, but you could alternatively set it to WIRELESS_MODE_ADHOC. Technically, all this is doing is setting the value of the variable to either 1 or 2, but that's not very self-explanatory so the defines that we set at the start of the sketchprovide easily memorable tokens.

Unsigned charwireless_mode=WIRELESS_MODE_INFRA;

The sketch then defines a couple of other variables for use by the WiShield.

unsigned char ssid_len;

unsigned char security_passphrase_len;

We also need to define some variables for processing the reading from the tank-level sensor. The sensorValue variable will hold the raw analog reading from the sensor and could have any value from 0 to 1023. The tankLevel variable will hold the tank level converted to a percentage. We have also defined some Boolean variable to keep track of the states of switches for whether to be on or off. Finally, we have defined the pin numbers that we used to control the digital output and the pin for analog input to measure the sensor as shown below.

```
intsensorValue = 0;        // value read from the pot
inttankLevel = 0;

boolean switch1State = false;
boolean switch2State = false;
booleanpumpSwitchState = false;

constint switch1Pin = 5;
constint switch2Pin = 6;
constintpumpSwitchPin = 7;
constintanalogInPin = 0;  // Analog input pin that the potentiometer is attached
to
```

The setup function is simple, but the WiServer.init() function is worth taking a look at. It accepts an argument that specifies the callback function to be executed inresponse to a connection request, and in this case we’ve told it to use the function sendWebPage(). This is a bit like setting up an interrupt because passing it to WiServer.init() it will be invoked automatically at the appropriate time.

void setup() {

WiServer.init(sendWebPage);

Next, the sketch opens a serial connection to the host so it can send status messages back to you, and enables "verbose" mode so the server will send log messages via that connection. We also setting our digital output pins.

```
Serial.begin(57600);
WiServer.enableVerboseMode(true);
pinMode(switch1Pin, OUTPUT);
pinMode(switch2Pin, OUTPUT);
pinMode(pumpSwitchPin, OUTPUT);
```

The main program loop is trivial. All it does is repeatedly call the WiServer .server_task() method so that incoming data queued by the WiShield will be processed. Without this, a connection request from your browser will arrive at the WiShield and sit in the buffer without ever being acted on. We have also implemented the some line of code to keep checking the water tank level.As you may notice that this code will read the sensor raw value and converted to a percentage value to represent the percentage of water in the tank. Whenever this percentage value decreased below the 10% it will turn the LED that represent the pump switch. If the level of the water reached 90%, the LED will turn off. Which mean the water pump is off.

```
void loop(){
// read the analog in value:
sensorValue = analogRead(analogInPin);
// map it to the range of the analog out:

tankLevel = map(sensorValue, 0, 1023, 0, 100);
if (tankLevel< 10){
pumpSwitchState = true;
digitalWrite(pumpSwitchPin, HIGH);
}
if (tankLevel> 90){
pumpSwitchState = false;
digitalWrite(pumpSwitchPin, LOW);
}
// Run WiServer

WiServer.server_task();
de
lay(10);
}
```

The sendWebPage() function generates the web page to send back to the browser. The function first check the URL to find out to control which switch. The state of the switch will toggle based on the URL that is send from the browser

```
booleansendMyPage(char* URL) {
if (strcmp(URL, "/switch1") == 0) switch1State = !switch1State;
if (strcmp(URL, "/switch2") == 0) switch2State = !switch2State;
if (strcmp(URL, "/pumpSwitch") == 0) pumpSwitchState =
!pumpSwitchState;
digitalWrite

(switch1Pin, switch1State);
digitalWrite(switch2Pin, switch2State);
digitalWrite(pumpSwitchPin, pumpSwitchState);
```

After that, the function will respond to the browser with the appropriate HTML code that the browser will understand to represent the page that will be the interface for our project.

```
WiServer.print("<html><center>");

WiServer.print("<h1>Switch Control</h1><br>");
```

```
printSwitchStatus("switch1", switch1State);
printSwitchStatus("switch2", switch2State);
printSwitchS
tatus("pumpSwitch", pumpSwitchState);
WiServer.print("<h2>Your water tank is ");
WiServer.print(tankLevel);
WiServer.print("&#37;");
WiServer.print(" full</h2>");
WiServer.print("<a href=/>Refresh Page</a>");
WiServer.prin
t("</center></html>");
```

Note we have also used another function called printSwitchStatus().
This function is used to simplify the code. And it is also used to print the state of all switch in the browser.

```
voidprintSwitchStatus(String switchName, booleanswitchState) {
WiServer.print(switchName);
WiServer.print(" is ");
if(switchState == false) {
WiServer.print(" <b>off</b><a href=/");
WiServer.print(switchName);
WiServer.print(">Turn On</a><br>");
} else {
WiServer.print(" <b>on</b><a href=/");
WiServer.print(switchName);
WiServer.print(">Turn off</a><br>");
}
}
```

CHAPTER 5: CONCLUSION

In conclusion, I feel that my product will be completed with the interfacing of the home appliances.

I explained the tools required to implement this project and the codes necessary for the Arduino board to create an interface with the home appliances it wants to control. The advantage of our product is that it is all based on Arduino which is an easy platform to implement a project of this type on. With this software, we can develop it for more complicated applications in the future. It could be implemented for appliances such as TV control, VCR setup and usage, stereo control, and even control of applications on the computer. In particular, we will provide installation instructions on setting up Home Surveillance.

REFERENCES

[1]Alfei, J. (2013). AutoHome: Using Arduino and `Smart' Android Apps to Remotely Control Appliances. 1st ed. whales.

[2] Engineersgarage.com. (2016). What is Home Automation | Introduction to Home Automation –EngineersGarage. [online] Available at: http://www.engineersgarage.com/articles/home-automation
[Accessed 12 May 2016]

[3] Gerhart, J. (1999). Home automation and wiring. New York: McGraw-Hill.

[4] Dave Rye Hometoys.com.(2016). Dave Rye @ X10 | HomeToys. [online] Available at:
http://www.hometoys.com/content.php?url=/htinews/oct99/articles/rye/rye.htm[Accessed 13 May 2016].

[5] Arduino.cc. (2016). Arduino -Introduction. [online] Available at:

https://www.arduino.cc/en/Guide/Introduction
[Accessed 14 May 2016].

[6] Adafruit Industries -Makers, hackers, artists, designers and engineers!.(2011). How many Arduinos are "in the wild?"About 300,000. [online]
Available at: https://blog.adafruit.com/2011/05/15/how-many-arduinos-are-in-the-wild-about-300000/[Accessed 12 May 2016].

[7] Medea. (2013). Arduino FAQ. [online] Available at: http://medea.mah.se/2013/04/arduino-faq/[Accessed 13 May 2016].

[8] GitHub. (2016). Optiboot/optiboot. [online] Available at: https://github.com/Optiboot/optiboot[Accessed 16 May 2016].

[9] Industries, A. (2016). DC Boarduino (Arduino compatible) Kit(w/ATmega328) [v1.0] ID: 72 -$17.50 : Adafruit Industries, Unique & funDIY electronics and kits. [online] Adafruit.com. Available at: https://www.adafruit.com/products/72[Accessed 19 May 2016].

[10] others, T. (2016). arduinosrl.it, Electronics. [online] Smartprj.com. Available at: http://smartprj.com/catalog/index.php
[Accessed 13 May 2016].

[11] Schmidt, M. (n.d.). Arduino.

[12] Programming Arduino getting started with sketches Mc@raw Hill.Aprl8,2016. Retrivied 28-3-2016

[13]" BlinkTutorial"Arduino.cc.

[14]" UsingAtmel studio for Arduino Development "Mequnolink.com

[15]" UsingAVR studio for Arduino development" Engbleze.com.Retrieved2016

[16] Arduino-noteBook_Ul-1.

[17] ParticleArduino cool projects for open Source Hardware

[18]By Jonathan oxer Hugh Blaming's