A graduate project submitted in partial fulfillment of the requirements for the degree of

Master of Science in Computer Engineering

By

Ehsan Dolatshahi

December 2015
The graduate project of Ehsan Dolatshahi is approved:

Prof. Somnath Chattopadhyay

Prof. Ramin Roosta

Prof. Nagi El Naga, Chair

California State University, Northridge
DEDICATION

This thesis is dedicated to my parents.

My lovely Mom and Dad

For their endless love, support and encouragement.

You made it possible

And

My beloved brother and sisters

For all motivation and for being the most supportive family in the world

You have been my inspiration each and every day

Thank you
ACKNOWLEDGEMENTS

Special thanks to

Dr. Nagi El Naga

My knowledgeable professor for his wise advices and help to finish this project.

I couldn't have done any of it without your support

Also

I would like to thank

All the CSUN professors and faculties of ECE department

For passing their knowledge during my graduate program
TABLE OF CONTENTS

| SIGNATURE PAGE                      | II |
| DEDICATION                          | III |
| ACKNOWLEDGEMENTS                    | IV |
| LIST OF FIGURES                     | VII |
| ABSTRACT                            | X  |

CHAPTER 1  INTRODUCTION  1

1.1 OVERVIEW  1
1.2 OBJECTIVES  4
1.3 REPORT OUTLINE  6

CHAPTER 2  SYSTEM DESCRIPTION  7

2.1 SYSTEM FUNCTIONS  7
2.2 SYSTEM BLOCK DIAGRAM  7
2.3 MICROPROCESSOR ORGANIZATION  9
2.4 ATmega2569 MICROPROCESSOR  12
2.5 SMART HOME AUTOMATION SYSTEM COMPONENTS  16
2.6 SYSTEM DESCRIPTION  34

CHAPTER 3  HARDWARE DESIGN  36

3.1 SYSTEM FUNCTIONS  36
3.2 ETHERNET SHIELD  43
3.3 PHOTORESISTOR  50
3.4 TEMPERATURE AND HUMIDITY SENSOR  52
3.5 GAS SENSOR  54
3.6 Human Body Pyroelectric Infrared Sensor (PIR) 56
3.7 RFID 59
3.8 Multi-Channel Relay 61

CHAPTER 4 SYSTEM SOFTWARE DESIGN 64

4.1 Introduction 64
4.2 Program Description 64
4.3 Subroutine Within the Main Routine 69
4.4 Subroutine Within the Network Interface 73
4.5 Photocell 76
4.6 Gas Sensor 76
4.7 Temperature and Humidity 79
4.8 Motion Detection Sensor (PIR) 79
4.9 RFID Module 82
4.10 Android Application Development 84

CHAPTER 5 CONCLUSION 87

5.1 System Evaluation 87
5.2 Conclusion 88

REFERENCES 89

APPENDIX A ARDUINO MICROCONTROLLER PROGRAM LIST 90
APPENDIX B ANDROID PROGRAM SETTING LIST 117
APPENDIX C DATASHEETS 119
# LIST OF FIGURES

1.1 General Scheme of a Smart Home Automation System 2

1.2 Connections and Devices of a Smart Home Automation System 3

2.1 System Block Diagram 8

2.2 Basic Computer Block Diagram 10

2.3 CPU Block Diagram 11

2.4 AVR CPU 13

2.5 ATmega2560 Microprocessor Block Diagram 14

2.6 Specifications of ATmega2560 15

2.7 ArduinoMEGA2560 16

2.8 Specifications of ArduinoMEGA2560 Microprocessor 16

2.9 Detail Components of an ArduinoMEGA2560 18

2.10 ArduinoMEGA2560 Reference Design 16

2.11 Photoresistor 20

2.12 Photoresistor Specifications 20

2.13 Temperature and Humidity Sensor (DHT11) 21

2.14 DHT11 Technical Specifications 21

2.15 MQ-2 Gas Sensor 22

2.16 MQ-2 Gas Sensor Specifications 22

2.17 PIR Sensor (Front) 24

2.18 PIR Sensor (Back) 24

2.19 RFID RC522 25

2.20 RFID Authorization Diagram 26

2.21 RFID Block Diagram 27

2.22 W5100 Ethernet Shield 29

2.23 W5100 Block Diagram 30

2.24 8-Channel Relay Module 31

2.25 4-Channel Relay Module 32
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.9</td>
<td>Flowchart for Time Out Subroutine</td>
<td>75</td>
</tr>
<tr>
<td>4.10</td>
<td>Flowchart of Delay Subroutine</td>
<td>75</td>
</tr>
<tr>
<td>4.11</td>
<td>Flowchart for Photocell Sensor</td>
<td>77</td>
</tr>
<tr>
<td>4.12</td>
<td>Flowchart for Gas Sensor</td>
<td>78</td>
</tr>
<tr>
<td>4.13</td>
<td>Flowchart for Temperature and Humidity Sensor</td>
<td>80</td>
</tr>
<tr>
<td>4.14</td>
<td>Flowchart for Motion Detection (PIR) Sensor</td>
<td>81</td>
</tr>
<tr>
<td>4.15</td>
<td>Flowchart for RFID Module</td>
<td>82</td>
</tr>
<tr>
<td>4.16</td>
<td>Flowchart for Android Application Interaction with Arduino Program</td>
<td>85</td>
</tr>
<tr>
<td>4.17</td>
<td>Flowchart for the Main Program Interaction with a Remote User</td>
<td>86</td>
</tr>
</tbody>
</table>
ABSTRACT

A SMART HOME AUTOMATION SYSTEM

By

Ehsan Dolatshahi

Master of Science in Computer Engineering

This graduate project presents a microprocessor-based home automation system. It demonstrates and explains the design and implementation of a digitally controlled interface between different sensors, RFID module and an Arduino microprocessor. This system could make a connection over Ethernet and send and receive data to a monitoring system. Also, it has a feature to control some part of the system remotely from any computer as well as a cellphone application.

The point of the system is automating the home to have a more convenient everyday life with a security feature to live safer.

This graduate project report discusses the basic concept of the design and presents circuit description in detail. In addition, it presents a basic concept of new microprocessor programming with the middle-class coding language as well as the basic concept of the most updated technologies of smart homes in the market which are controllable over the internet and smart phones.

Finally, this report tries to emphasize the point of advantages of new microprocessor technologies which could make many more houses smart with lower cost. Therefore, the middle class could afford them more easily and enjoy greater convenience and safety.
CHAPTER 1

INTRODUCTION

1.1. Overview

It is a fact that human beings have been creating everything to live more conveniently since the beginning of the world. Automating works and objects is one of the major topics which is helping to achieve this fact. Thanks to the technology, these days we are going toward automating everything to achieve comfort as much as we can. Since people spend a big portion of their life at home, automating a house would be an attractive subject for ease of life.

With a home automation system, people are able to program many objects to work automatically, which is automation of the home, housework or household activities. In addition, there are new available features that let users control objects remotely. Nowadays most people have their smartphone nearby them; therefore, adding an interface on the smartphone to control an automated system is a big plus.

Home automation systems could control lighting system, heating and air conditioning, appliance and security door locks to bring more accessibility, convenience, safety, ease and energy efficiency.

An automated home is an integrated system to gather security systems and appliances. This house system is connectable to a smart grid system over a network, which causes energy saving as well as decreasing the cost.
The new idea of “Internet of Things” (IoT) is growing fast and people are doing more and more work over the Internet. All controllable objects in a house could be connected into a local area network (LAN) and the main board of the system also could send and receive data over the internet for a remote user. Both of the local connection and the mainboard to the internet connection could be wired or wireless with different technologies such as WLAN, ZigBee (XBee), Bluetooth, etc.
Figure 1.2 Connections and Devices of a Smart Home Automation System
Security is an important subject that could be added to a home automation system. Gathering all different security systems such as alarm, the fire alarm, access control (door lock), CCTV, intrusion detection system (IDS), etc. makes our life safer. All the security systems mentioned above could be integrated into a home automation system.

Energy saving is another benefit of the smart home system. It enables us to save energy and cost with having a smart control of the heating and air conditioning system, lighting system, water sprinkler system, solar system, etc.

1.2. Objectives

The objective of this project is to use Arduino Mega2560 microcontroller to design and build a smart home automation system which provides the user with new features such as door access control, smart lighting system, fire alarm system and heating/cooling system plus remote access to control home appliance and objects wirelessly over Ethernet via any computer or smartphone. The purpose of the project is to bring comfort, security and energy saving to our lives.

- Comfort:
  - Automatic and programmable heating/cooling system, so no need to turn on/off manually. This feature could be utilized with:
    - Temperature detection sensor
  - Automatic and programmable lighting system using two different technologies:
    - Light sensor
    - Motion detection sensor
- Access control: to open the door lock with a programmed tag issued for different tag holders by scanning tags from:
  - RFID (Radio-frequency Identification) tag reader module

- Remote access to turn on/off home objects over network via:
  - Any computer’s browser
  - Smartphone application

- Energy saving:
  - To keep the room at a preferred temperature, the smart system would control heating/cooling system in a programmed mode versus working all the time or manually. The module facilitating this feature is:
    - Temperature sensor
  - The smart lighting system would save energy and cost with turning on/off the lights based on the lighting level of the room. Also even in the dark it detects motions to turn on the light and it also could have a timer setup.
  - Turning off the unnecessary working home appliances remotely such as turning off a TV from work which one has forgotten to turn off.

- Security/safety:
  - Access control to get to the house only to those who are tag holders.
  - Close the garage door remotely in case one forgot when leaving the house.
1.3. Report Outline

The development of this system has been described in five chapters:

<table>
<thead>
<tr>
<th>Chapter II</th>
<th>Describes system function and the basic element of a microcomputer system, the detail of the Arduino microcontroller and all different I/O port.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chapter III</td>
<td>Describes the different blocks which the system is made of, and the detail of the Arduino microprocessor-based computer system and the detailed circuit design including all different modules used in this project.</td>
</tr>
<tr>
<td>Chapter IV</td>
<td>Presents the software programs, codes and techniques used to implement the system. This chapter also includes some of the details of the smartphone application programming.</td>
</tr>
<tr>
<td>Chapter V</td>
<td>Presents the evaluation of the system performance and the result of connections.</td>
</tr>
</tbody>
</table>
CHAPTER 2

SYSTEM DESCRIPTION

2.1. System Functions

The following functions are performed by the system circuit:

1. Signal the microprocessor when any module sends data to it
2. Convert analog inputs
3. Process the input data
4. Communicate with the Ethernet module to send and receive data
5. Convert to the data output
6. Send output signal to output modules

Also, the smartphone application allows users to send demands wirelessly over the network, then commands will be issued by microcontroller toward relays. The cell phone application could operate eight control on/off switches simultaneously, such as opening/closing curtains, turning on/off A/C, lighting, stereo, access-control systems and more.

2.2. System Block Diagram

Based on the design requirements and specifications, the system block diagram shown on Figure 2.1 is developed. This block diagram defines all the function to be performed by the system. A modular approach to project design was taken. The system is designed based on an Arduino Mega2560 microcontroller which is based on an Atmel2560 microprocessor.

In the following section, some of the basic concepts of circuits that are used in the system design are explained.
Figure 2.1 System Block Diagram
2.3. Microcomputer Organization

A microcomputer is a device that operates on binary information. A typical microcomputer consists of:

1) A Central Processor Unit (CPU)
2) A Memory (RAM and ROM)
3) Input/output (I/O) ports

All these units communicate to each other through a bus structured organization as shown in Figure 2.2.

1) The CPU brings the entire system together. The function implemented by the other elements is under CPU control. Also, all instructions from memory will be fetched by CPU. Therefore all the binary substances will be decoded and become executive. A typical CPU contains the following interconnected functional units: (Figure 2.3)

   i. Memory Unit (Registers)
      Temporary storage unit in the CPU

   ii. Arithmetic/Logic Unit (ALU)
      The Main core of the CPU. All the logical and arithmetic processes on binary data will be accomplished by ALU

   iii. Control Unit
      Make a use of clock inputs, CU keeps the appropriate arrangement of proceedings essential for all processing tasks
Basic computer system consist of a Central processing unit (CPU), memory (RAM and ROM), input/output (I/O) unit.

Figure 2.2 Basic Computer Block Diagram
2) Memory: Microprocessors need to save/restore data for all processing programs on/from a memory unit. RAM and ROM are two different type of memory in the computer systems.

i. RAM: (Random Access Memory) A type of memory which is readable and writeable by the processor. Every time the system power goes off, this memory will be all cleared.

ii. ROM (Read Only Memory): It is not a writable type of memory, so only the processor is able to read data from it. Since ROM is a memory with fixed
instructions, the instructions will be saved into the ROM even when the system power is off.

3) I/O ports: Input and output ports to make communications between all different components of the computer system with sending/receiving data or signals.

2.4. ATmega2560 microprocessor

This microprocessor is manufactured by Atmel company which is a low-power CMOS 8-bit microcontroller based on the AVR enhanced (Figure 2.4) RISC architecture, which consists of 256KB Flash, 4KB of EEPROM, 8KB of RAM, 86 General Purpose I/O pins, 12 of 16bits resolution PWN channels, 4 Serial USARTs, and 16 ADC Channels. Figure 2.5 presents the ATmega2560 microprocessor block diagram and Figure 2.6 presents the ATmega2560 microprocessor specifications.
Figure 2.4 AVR CPU
Figure 2.5 ATmega2560 Microprocessor Block Diagram
### Specifications of ATMEGA2560

<table>
<thead>
<tr>
<th>Feature</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flash (kbytes)</td>
<td>256 Kbytes</td>
</tr>
<tr>
<td>Max. Operating Frequency</td>
<td>16 MHz</td>
</tr>
<tr>
<td>Hardware Qtouch Acquisition</td>
<td>No</td>
</tr>
<tr>
<td>Ext Interrupts</td>
<td>32</td>
</tr>
<tr>
<td>Usb Interface</td>
<td>No</td>
</tr>
<tr>
<td>Twi (I2c)</td>
<td>1</td>
</tr>
<tr>
<td>Graphic Lcd</td>
<td>No</td>
</tr>
<tr>
<td>Camera Interface</td>
<td>No</td>
</tr>
<tr>
<td>Adc Resolution (bits)</td>
<td>10</td>
</tr>
<tr>
<td>Analog Comparators</td>
<td>1</td>
</tr>
<tr>
<td>Temp. Sensor</td>
<td>No</td>
</tr>
<tr>
<td>Sram (kbytes)</td>
<td>8</td>
</tr>
<tr>
<td>Self Program Memory</td>
<td>YES</td>
</tr>
<tr>
<td>Pin Count</td>
<td>100</td>
</tr>
<tr>
<td>Cpu</td>
<td>8-bit AVR</td>
</tr>
<tr>
<td>Max I/o PIns</td>
<td>86</td>
</tr>
<tr>
<td>Usb Speed</td>
<td>No</td>
</tr>
<tr>
<td>Spi</td>
<td>5</td>
</tr>
<tr>
<td>Uart</td>
<td>4</td>
</tr>
<tr>
<td>Video Decoder</td>
<td>No</td>
</tr>
<tr>
<td>Adc Channels</td>
<td>16</td>
</tr>
<tr>
<td>Adc Speed (ksp/s)</td>
<td>15</td>
</tr>
<tr>
<td>Resistive Touch Screen</td>
<td>No</td>
</tr>
<tr>
<td>Crypto Engine</td>
<td>No</td>
</tr>
<tr>
<td>Eeprom (bytes)</td>
<td>4095</td>
</tr>
<tr>
<td>Dram Memory</td>
<td>No</td>
</tr>
</tbody>
</table>

Figure 2.6 Specifications of ATMEGA2560
2.5. Smart Home Automation System Components

This project is using an Arduino MEGA2560 (Figure 2.7), which is based on an ATmega2560 microprocessor.

![Arduino MEGA2560](image)

Figure 2.7 ArduinoMega2560

2.5.1. Arduino MEGA2560 Microcontroller

An Arduino MEGA2560 Microcontroller has been designed based on an ATmega2560 microprocessor that runs at the speed of 16MHz. As Figure 2.8 shows below, it contains 54 digital input/output pins, 14 of them could be used as PWM (Pulse Width Modulation is a method for getting analog results with digital means) outputs. Furthermore, it contains 16 analog inputs and 4 hardware serial ports.
2.5.1.1. Arduino MEGA2560 Components

As shown in Figure 2.9, a Mega2560 microcontroller has 54 digital I/O pins, any of which could be used either for input or output. They all work at 5 volts and each of them is able to provide/receive a maximum 40 mA. Moreover, they have an inner pull-up resistor 20 to 50 kΩ. This microcontroller has 16 analog inputs. There is a reset button on the board to reset the microprocessor. Also, it has a test LED, A USB compatible connection port, a 9V power jack including an ICSP header which is In-Circuit Serial Programming header. MEGA2560 supports PC (Inter-Integrated Circuit) communication. Arduino MEGA2560 reference design is shown in Figure 2.10
Figure 2.9 Detail Components of an Arduino MEGA2560
Figure 2.10 Arduino MEGA2560 Reference Design
2.5.2. Input Components

In this project, we have used photoresistor (light sensor), Temperature and Humidity sensor, Gas sensor, and Motion detection sensor and RFID tag reader for authentication.

2.5.2.1. Sensors

i. Photoresistor:

It is a light dependent resistor (LDR). The resistance of a photoresistor (Figure 2.11) decreases with increasing incident light intensity of the light source. When the light level increases the resistance of photoresistor decreases, and when lighting level is not high enough it will have a higher resistance. This analog sensor needs a voltage divider in order to receive an accurate signal. Photoresistor specification is available in Figure 2.12.

![Photoresistor](image)

Figure 2.11 Photoresistor

<table>
<thead>
<tr>
<th>Specification</th>
<th>Light resistance (10Lux) (KΩ)</th>
<th>Dark resistance (MΩ)</th>
<th>γ (10^6)</th>
<th>Response time (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>4-7</td>
<td>0.5</td>
<td>0.5</td>
<td>Increase: 30</td>
</tr>
</tbody>
</table>

Figure 2.12 Photoresistor Specifications
ii. Temperature and Humidity sensor (DHT11):

This sensor (Figure 2.13) measures the temperature in Celsius /Fahrenheit. Also, it is able to measure humidity. See the detail technical specification in Figure 2.14.

![Temperature and Humidity Sensor (DHT11)](image)

Figure 2.13 Temperature and Humidity Sensor (DHT11)

<table>
<thead>
<tr>
<th>Item</th>
<th>Measurement Range</th>
<th>Humidity Accuracy</th>
<th>Temperature Accuracy</th>
<th>Resolution</th>
<th>Package</th>
</tr>
</thead>
<tbody>
<tr>
<td>DHT11</td>
<td>20-90%RH</td>
<td>±5%RH</td>
<td>±2°C</td>
<td>1</td>
<td>4 Pin Single Row</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Conditions</th>
<th>Minimum</th>
<th>Typical</th>
<th>Maximum</th>
</tr>
</thead>
<tbody>
<tr>
<td>Humidity</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Resolution</td>
<td></td>
<td>1%RH</td>
<td>1%RH</td>
<td>1%RH</td>
</tr>
<tr>
<td>Repeatability</td>
<td></td>
<td>±1%RH</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Accuracy</td>
<td></td>
<td>±4%RH</td>
<td></td>
<td>±5%RH</td>
</tr>
<tr>
<td>Interchangeability</td>
<td></td>
<td>Fully Interchangeable</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Measurement Range</th>
<th>Measurement Range</th>
<th>Measurement Range</th>
<th>Measurement Range</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0°C</td>
<td>30%RH</td>
<td>90%RH</td>
</tr>
<tr>
<td></td>
<td>25°C</td>
<td>20%RH</td>
<td>90%RH</td>
</tr>
<tr>
<td></td>
<td>50°C</td>
<td>20%RH</td>
<td>80%RH</td>
</tr>
</tbody>
</table>

| Response Time (Seconds) | 1/e(63%) 25°C, 1m/s Air | 6S | 10S | 15S |

2.14 DHT11 Technical Specifications
iii. Gas Sensor (MQ-2):

SnO2 (oxide semiconductor) is the responsive object of the MQ-2 gas sensor (Figure 2.15). Once SnO2 recognizes the gas existence, the sensor conductivity of the module will be increased along with the gas absorption. Then, it adapts variation of conductivity as well as the output signal of gas absorption. MQ-2 is sensitive to detect propane, natural gas, LPG, and other smokes. Detailed MQ-2 specifications have shown in Figure 2.16.

Figure 2.15 MQ-2 Gas Sensor
<table>
<thead>
<tr>
<th>Character</th>
<th>Model No.</th>
<th>MQ-2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sensor Type</td>
<td>Semiconductor</td>
<td></td>
</tr>
<tr>
<td>Standard Encapsulation</td>
<td>Bakelite (Black Bakelite)</td>
<td></td>
</tr>
<tr>
<td>Detection Gas</td>
<td>Combustible gas and smoke</td>
<td></td>
</tr>
<tr>
<td>Concentration</td>
<td>300-10000ppm (Combustible gas)</td>
<td></td>
</tr>
<tr>
<td>Circuit</td>
<td>Loop Voltage</td>
<td>$V_c$</td>
</tr>
<tr>
<td></td>
<td>Heater Voltage</td>
<td>$V_H$</td>
</tr>
<tr>
<td></td>
<td>Load Resistance</td>
<td>$R_L$</td>
</tr>
<tr>
<td></td>
<td>Heater Resistance</td>
<td>$R_H$</td>
</tr>
<tr>
<td></td>
<td>Heater consumption</td>
<td>$P_H$</td>
</tr>
<tr>
<td></td>
<td>Sensing Resistance</td>
<td>$R_s$</td>
</tr>
<tr>
<td></td>
<td>Sensitivity</td>
<td>$S$</td>
</tr>
<tr>
<td></td>
<td>Slope</td>
<td>$\alpha$</td>
</tr>
<tr>
<td>Condition</td>
<td>Tem. Humidity</td>
<td>$20^\circ C \pm 2^\circ C$; $65% \pm 5%$RH</td>
</tr>
<tr>
<td></td>
<td>Standard test circuit</td>
<td>$V_c$: $5.0V \pm 0.1V$; $V_H$: $5.0V \pm 0.1V$</td>
</tr>
<tr>
<td></td>
<td>Preheat time</td>
<td>Over 48 hours</td>
</tr>
</tbody>
</table>

Figure 2.16 MQ-2 Gas Sensor Specifications
iv. Human Body Pyroelectric Infrared Sensor (PIR)

This sensor (Figure 2.17 and 2.18) detects any change in infrared radiation. Therefore, it could be used to detect motion. PIR’s IC is BISS0001 Micro Power, which uses low-power CMOS technology.

Figure 2.17 PIR Sensor (Front)

Figure 2.18 PIR Sensor (Back)
2.5.2.2. RFID RC522:

As the name applies RFID stands for radio frequency identification. RFID (Figure 2.19) is a module to transmit data in a short range (~4 inches) in 13.56 MHz frequency. This module includes a tag scanner/reader and it is capable of scanning as many tags as you set to it. Each tag has a specific ID to be set into the programming part to be determined authorized or not authorized. Transferring data rate is 106KB per second, which is 100% ASK (Amplitude Shift Keying).

![Figure 2.19 RFID RC522](image)

Also, there is a master key to program the RFID module faster (Figure 2.20). If the master key gets scanned first and then an unknown key, the unknown key will become an authorized tag. However, if the master key gets scanned first and then a known key, the known key will become an unauthorized key. The RFID module block diagram has shown in Figure 2.21
Figure 2.20 RFID Authorization Diagram
Figure 2.21 RFID Block Diagram
2.5.2.3. Ethernet Shield (W5100)

In this project, we have utilized W5100 network interface (Figure 2.22) which uses 0.18 µm CMOS technology and has 16Kbytes of internal memory (TX/RX buffer) to provide a 10Mb/100Mb Ethernet connection. It is able to connect to the internet without an OS (Operation System). It is well compatible with IEEE802.3/u standard, TCP/IP protocol and the MAC addressing feature, which makes this module useful for intergrading with many embedded systems. The W5100 has a micro SD interface and an Ethernet socket (RJ-45), which supports the power over Ethernet (POE) feature. W5100 block diagram has shown in Figure 2.23.

W5100 TCP protocol supports:

- Transmission Control protocol (TCP)
- User Datagram Protocol (UDP)
- Internet Protocol Version 4 (IPV4)
- Internet Control Message Protocol (ICMP)
- Address Resolution Protocol (ARP)
- Internet Group Massaging Protocol (IGMP)
- Point-to-Point Protocol over Ethernet (PPPoE).
Figure 2.22 W5100 Ethernet Shield
Figure 2.23 W5100 Block Diagram
2.5.2.4. Multi-Channel Relay Board

With this module, we have utilized 12 relays for switching application, which is the total of one 8-Channel relay module (Figure 2.24) and one 4-Channel relay module (Figure 2.25). They both have the same technology, therefore, we only show the circuit detail of the 8-Channel relay. Input voltage could be TTL or CMOS, which works with AC directly for home automation purposes. The AC supply could switch any appliance such as fan, light, intrusion detection alarm, fire alarm, etc. Figure 2.26 shows the electrical circuit for a single relay channel and Figure 2.27 has detailed information for an 8-Channel relay board.

![Figure 2.24 8-Channel Relay Module](image)
Figure 2.25 4-Channel Relay Module

Figure 2.26 Electrical Circuit Diagram for a Single Channel Relay
Figure 2.27 8-Channel Relay Circuit Diagram
2.6. System Description

Besides the Arduino MEGA2560 microprocessor-based microcomputer used in this system, which can be called the brain of the system, the network interface (W5100), sensors, and the RFID module also serve a very important role in our system. (The detailed hardware design description for all modules is presented in chapter 3).

The network interface used in this project not only communicates with the computer in client side for transferring sensor’s data, but also it communicates with the server to show the status of each relay. It allows the user to control the 8-Channle relay and consequently all connected appliances to those relays, which could be controlled over the network. A mobile application has been designed for smartphones to provide an easier way to control those relays remotely.

The RFID tag reader is always in the waiting status for giving entrance access to authorized users for convenience and security purposes. Moreover, the smart home system allows the user to add a keyless home entry feature in order to open the door/gate remotely. The keyless home entry is controllable by a cell phone application and any phone/computer’s browser which could gain access to the home area network (HAN) directly or over the internet.

All sensors communicate with a microprocessor, then it processes all input data, and based on the program, the microprocessor makes the decision and sends signals to different relays in order to automate the home appliances. Therefore, the temperature sensor collaborates with a relay that is connected to a fan or an air-conditioning system. The gas detection sensor collaborates with a relay that is connected to a fire alarm system (smoke detection system). The light sensor
collaborates with a relay which is connected to a lighting system, and the motion detector sensor collaborates with a relay which is connected to an alarm system.

Furthermore, this system is able to get more add-on sensors and get integrated with more modules such as solar system and smart grid system.

In this project, we have provided the basic concept of the smart home automation, which is growing every day and we have demonstrated the ability of this system to get expanded and intergraded as an up-to-date embedded system.
3.1. Computer Hardware Design

The computer is designed based on an Arduino MEGA2560 microcontroller. The basic components consist of buffers, decoder, RAM, ROM, and I/O ports. The diagram is shown in Figure 2.5.

Figure 3.1 shows Arduino MEGA2560 microcontroller pin mapping scheme. In order to have a better visual, a magnified capture of each side has been shown individually in 3.1.1 through 3.1.4

* Note: In Figure 3.1 and 3.1.1 through 3.1.4:
  - Red writing are the mapped by the pin number on Arduino board.
  - Black writing are the name of the pin on CPU
  - Black numbers in boxes are the CPU pin number.

Figure 3.2 shows the connection between CPU pins (1<sup>st</sup> column) to the Arduino MEGA 2560 board (3<sup>rd</sup> column). Also, the connection between each pin of Arduino (3<sup>rd</sup> column) and each module (4<sup>th</sup> column) is presented in this Figure.

As shown in the table, pin 31 of the CPU is connected to VCC pin of Arduino and pin 32 of the CPU is connected to the GND of Arduino.
Figure 3.1 MEGA2560 Pin Mapping Scheme
<table>
<thead>
<tr>
<th>Digital pin 4 (PWM)</th>
<th>PG5 (OC0B)</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Digital pin 0 (RX0)</td>
<td>PE0 (RXD0/PCINT8)</td>
<td>2</td>
</tr>
<tr>
<td>Digital pin 1 (TX0)</td>
<td>PE1 (TXD0)</td>
<td>3</td>
</tr>
<tr>
<td>Digital pin 5 (PWM)</td>
<td>PE2 (XCK0/AIN0)</td>
<td>4</td>
</tr>
<tr>
<td>Digital pin 2 (PWM)</td>
<td>PE3 (OC3A/AIN1)</td>
<td>5</td>
</tr>
<tr>
<td>Digital pin 3 (PWM)</td>
<td>PE4 (OC3B/INT4)</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>PE5 (OC3C/INT5)</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td>PE6 (T3/INT6)</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td>PE7 (CLKO/ICP3/INT7)</td>
<td>9</td>
</tr>
<tr>
<td>VCC</td>
<td>VCC</td>
<td>10</td>
</tr>
<tr>
<td>GND</td>
<td>GND</td>
<td>11</td>
</tr>
<tr>
<td>Digital pin 17 (RX2)</td>
<td>PH0 (RXD2)</td>
<td>12</td>
</tr>
<tr>
<td>Digital pin 16 (TX2)</td>
<td>PH1 (TXD2)</td>
<td>13</td>
</tr>
<tr>
<td></td>
<td>PH2 (XCK2)</td>
<td>14</td>
</tr>
<tr>
<td>Digital pin 6 (PWM)</td>
<td>PH3 (OC4A)</td>
<td>15</td>
</tr>
<tr>
<td>Digital pin 7 (PWM)</td>
<td>PH4 (OC4B)</td>
<td>16</td>
</tr>
<tr>
<td>Digital pin 8 (PWM)</td>
<td>PH5 (OC4C)</td>
<td>17</td>
</tr>
<tr>
<td>Digital pin 9 (PWM)</td>
<td>PH6 (OC2B)</td>
<td>18</td>
</tr>
<tr>
<td>Digital pin 53 (SS)</td>
<td>PB0 (SS/PCINT0)</td>
<td>19</td>
</tr>
<tr>
<td>Digital pin 52 (SCK)</td>
<td>PB1 (SCK/PCINT1)</td>
<td>20</td>
</tr>
<tr>
<td>Digital pin 51 (MOSI)</td>
<td>PB2 (MOSI/PCINT2)</td>
<td>21</td>
</tr>
<tr>
<td>Digital pin 50 (MISO)</td>
<td>PB3 (MISO/PCINT3)</td>
<td>22</td>
</tr>
<tr>
<td>Digital pin 10 (PWM)</td>
<td>PB4 (OC2A/PCINT4)</td>
<td>23</td>
</tr>
<tr>
<td>Digital pin 11 (PWM)</td>
<td>PB5 (OC1A/PCINT5)</td>
<td>24</td>
</tr>
<tr>
<td>Digital pin 12 (PWM)</td>
<td>PB6 (OC1B/PCINT6)</td>
<td>25</td>
</tr>
</tbody>
</table>

Figure 3.1.1 MEGA2560 Pin Mapping (Left Part)
Figure 3.1.2 MEGA2560 Pin Mapping (Top Part)
Figure 3.1.3 MEGA2560 Pin Mapping (Right Part)
Figure 3.1.4 MEGA2560 Pin Mapping (Bottom Part)
<table>
<thead>
<tr>
<th>CPU Pin #</th>
<th>Pin Name (On CPU)</th>
<th>Mapped Pin Name (On Arduino)</th>
<th>Connected Module</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>VCC</td>
<td>VCC</td>
<td>VCC</td>
</tr>
<tr>
<td>32</td>
<td>GND</td>
<td>GND</td>
<td>GND</td>
</tr>
<tr>
<td>97</td>
<td>PF0 (ADC0)</td>
<td>Analog pin 0</td>
<td>Photoresistor</td>
</tr>
<tr>
<td>96</td>
<td>PF1 (ADC1)</td>
<td>Analog pin 1</td>
<td>Gas Sensor</td>
</tr>
<tr>
<td>6</td>
<td>PE4 (OC3B/INT4)</td>
<td>Digital pin 2 (PWM)</td>
<td>RFID IRQ</td>
</tr>
<tr>
<td>7</td>
<td>PE5 (OC3C/INT5)</td>
<td>Digital pin 3 (PWM)</td>
<td>RFID SCK</td>
</tr>
<tr>
<td>1</td>
<td>PG5 (OC0B)</td>
<td>Digital pin 4 (PWM)</td>
<td>RFID MOSI</td>
</tr>
<tr>
<td>5</td>
<td>PE3 (OC3A/AIN1)</td>
<td>Digital pin 5 (PWM)</td>
<td>RFID MISO</td>
</tr>
<tr>
<td>15</td>
<td>PH3 (OC4A)</td>
<td>Digital pin 6 (PWM)</td>
<td>RFID NSS</td>
</tr>
<tr>
<td>16</td>
<td>PH4 (OC4B)</td>
<td>Digital pin 7 (PWM)</td>
<td>RFID RST</td>
</tr>
<tr>
<td>56</td>
<td>PC3 (A11)</td>
<td>Digital pin 34</td>
<td>Temperature Sensor</td>
</tr>
<tr>
<td>54</td>
<td>PC1 (A9)</td>
<td>Digital pin 36</td>
<td>Motion Sensor</td>
</tr>
<tr>
<td>50</td>
<td>PD7 (T0)</td>
<td>Digital pin 38</td>
<td>LED to motion</td>
</tr>
<tr>
<td>52</td>
<td>PG1 (RD)</td>
<td>Digital pin 40</td>
<td>LED to RFID</td>
</tr>
<tr>
<td>42</td>
<td>PL7</td>
<td>Digital pin 42</td>
<td>Relay 1 of 4</td>
</tr>
<tr>
<td>40</td>
<td>PL5 (OC5C)</td>
<td>Digital pin 44 (PWM)</td>
<td>Relay 2 of 4</td>
</tr>
<tr>
<td>38</td>
<td>PL3 (OC5A)</td>
<td>Digital pin 46 (PWM)</td>
<td>Relay 3 of 4</td>
</tr>
<tr>
<td>36</td>
<td>PL1 (ICP5)</td>
<td>Digital pin 48</td>
<td>Relay 4 of 4</td>
</tr>
<tr>
<td>59</td>
<td>PC6 (A14)</td>
<td>Digital pin 31</td>
<td>Relay 1 of 8</td>
</tr>
<tr>
<td>57</td>
<td>PC4 (A12)</td>
<td>Digital pin 33</td>
<td>Relay 2 of 8</td>
</tr>
<tr>
<td>55</td>
<td>PC2 (A10)</td>
<td>Digital pin 35</td>
<td>Relay 3 of 8</td>
</tr>
<tr>
<td>53</td>
<td>PC0 (A8)</td>
<td>Digital pin 37</td>
<td>Relay 4 of 8</td>
</tr>
<tr>
<td>70</td>
<td>PG2 (ALE)</td>
<td>Digital pin 39</td>
<td>Relay 5 of 8</td>
</tr>
<tr>
<td>51</td>
<td>PG0 (WR)</td>
<td>Digital pin 41</td>
<td>Relay 6 of 8</td>
</tr>
<tr>
<td>41</td>
<td>PL6</td>
<td>Digital pin 43</td>
<td>Relay 7 of 8</td>
</tr>
<tr>
<td>39</td>
<td>PL4 (OC5B)</td>
<td>Digital pin 45 (PWM)</td>
<td>Relay 8 of 8</td>
</tr>
</tbody>
</table>

Figure 3.2 CPU Pins Connection to Arduino MEGA2560 and Arduino Pins Connection to Each Module
3.2. Ethernet Shield (W5100)

Network interface (W5100) circuit schematic in Figure 3.3 shows how each pin of the main chip has been connected to the other part of the circuit. In Figure 3.3.1 through 3.3.4, we can see the detailed circuit design for each part of the W5100. Pins D0 to D7 are for data transmitting.

SPI (Serial Peripheral Interface) is a data protocol utilized by microcontrollers to communicate with single or multiple peripheral devices fast in a short range. It could also be used for interaction among two microcontrollers. Usually, there are SCK, MOSI, and MISO lines mutual to all these devices.

SCK (Serial Clock) are the clock pulses that coordinate data communication made by the main and single pin point to each device. MOSI (Master Out Slave In) is the main pin for transferring data to the other devices. MISO (Master In Slave Out) is the Slave pin for transferring data to the main. ISP/ICSP (In-system Programming/In-Circuit Serial Programming) is the capability of microcontrollers to be programmed while installed in a complete system, instead of obliging the chip to be programmed before positioning it into the system.

Figure 3.4 shows how MISO, SCK, Reset, GND, VCC, and MOSI pins connect to ICSP/ISP headers on Mega2560 shown in Figure 3.5.

As it has shown in below pin 9, 33, 34 of AVR CPU are connected to ISP headers.

<table>
<thead>
<tr>
<th>AVR pin #</th>
<th>AVR pin name</th>
<th>Arduino ISCP header Mapped pin name</th>
<th>Connected Module</th>
</tr>
</thead>
<tbody>
<tr>
<td>#9</td>
<td>PE7 (CLK0/ICP3/INT7)</td>
<td></td>
<td>W5100 MOSI</td>
</tr>
<tr>
<td>#33</td>
<td>(XTAL2)</td>
<td>(XTAL2)</td>
<td>W5100 MISO</td>
</tr>
<tr>
<td>#34</td>
<td>(XTAL1)</td>
<td>(XTAL1)</td>
<td>W5100 SCK</td>
</tr>
<tr>
<td></td>
<td>Reset</td>
<td>Reset</td>
<td>W5100 Reset</td>
</tr>
<tr>
<td></td>
<td>GND</td>
<td>GND</td>
<td>W5100 GND</td>
</tr>
<tr>
<td></td>
<td>VCC</td>
<td>VCC</td>
<td>W5100 VCC</td>
</tr>
</tbody>
</table>
Figure 3.3 W5100 Circuit Schematic
To use the SPI interface, pull up the SEN pin. Otherwise pull down the SEN. When using SPI mode, pull down the ADDR pins.

Figure 3.3.1 W5100 Circuit Schematic
Figure 3.3.2 W5100 Circuit Schematic
Place C6, C7, FB2 close to 1V8OUT and place C8 close to 1V8A

Place C9, FB3 close to 3V3D and place C10, C11 close to 3V3A

Figure 3.3.3 W5100 Circuit Schematic
Figure 3.3.4 W5100 Circuit Schematic
Figure 3.4 W5100 Pin Mapping Schematic
3.3. Photoresistor:

Photoresistor module has three pins (Figure 3.6). Pin A is connected to analog pin 0 (A0) of Arduino, pin B is concocted to 5V (VCC) and C is connected to GND. The Arduino’s analog pin 0 (A0) is concocted to PF0 (ADC0) pin of AVR CPU which is pin 97. Pin A0 is reading the analog data from photoresistor sensor. Figure 3.7 shows the diagram of the connection between a photoresistor and the Arduino microprocessor.
Figure 3.6 A Photoresistor Module Soldered to a Board

Figure 3.7 The Connection of a Photoresistor to the Arduino microcontroller
3.4. Temperature and Humidity Sensor (DHT-11)

The digital output of this sensor transfers (8 bit RH) data through the signal pin shown in Figure 3.8. The signal pin is connected to the digital pin 34 of Arduino (Figure 3.9), which is connected to the pin 56/PC3 (A11) of AVR CPU. A 5k pull-up resistor is recommended for a connection cable shorter than 20 meters.

![Figure 3.8 Temperature and Humidity Sensor (DHT-11)](image)
Figure 3.9 Temperature and Humidity Sensor (DHT-11) Connected to the Arduino.
3.5. Gas Sensor (MQ-2)

In the Gas sensor MQ-2 module, there are three pins (Figure 3.10). Pin 1 sends analog data from pin 1 in (Figure 3.10) to analog input pin 1 (A1) of the Arduino (Figure 3.11) board, which is connected to pin 96 PF1(ADC) on AVR CPU. Pin 2 of MQ-2 is connected to the VCC and pin 3 is connected to the GND. The sensitivity of the Gas detection sensor can be fine-tuned by the potentiometer (Sensing Resistance range: 1KΩ 50ppm toluene to 20KΩ in.).

Figure 3.10. Gas Sensor Module (MQ-2) Circuit Diagram
Figure 3.11 Gas Sensor Module (MQ-2) Connection to an Arduino Board.
3.6. Human Body Pyroelectric Infrared Sensor (PIR)

This module, also called a motion detection sensor works with Infra-red technology. PIR has three pins (Figure 3.12): pin A is connected to the GND, pin C is connected to the VCC, and pin B is the digital output signal and it is connected to the Digital input pin 36 of Arduino (Figure 3.13), which is connected to pin 54 PC1 (A9) of AVR CPU. Digital pin 38 of Arduino is an output for an LED that turns on once the sensor detects any motion. Arduino digital pin 38 is connected to pin 50 PD7 (T0) of AVR CPU. Figure 3.14 shows the circuit diagram of a PIR sensor.

![Figure 3.12 Human Body Pyroelectric Infrared Sensor (Back View) and its Pins](image-url)
Figure 3.13 Scheme of Connection between A PIR Sensor Module and the Arduino
3.7. RFID (RC522)

Once an RFID (Radio Frequency Identification) reader module scans a tag, it sends the digital signals to the Arduino over five pins (D, E, F, G, H). As Figure 3.15 shows, Pin A is connected to the VCC; pin B (RST) is connected to the Digital pin 7 of Arduino, which is connected to pin 16 of AVR CPU; and pin C in connected to the GND. In addition, pin D (MISO) is connected to the Digital pin 5 (PWM) of Arduino, which is connected to pin 5 of AVR CPU; pin E (MOSI) is connected to the Digital pin 4 (PWM) of Arduino, which is connected to pin 1 of AVR CPU; pin F (SCK) is connected to the Digital pin 3 (PWM) of Arduino, which is connected to pin 7 of AVR CPU; pin G (NSS) is connected to the Digital pin 6 (PWM) of Arduino, which is connected to pin 15 of AVR CPU; and pin H (IRQ) is connected to the Digital pin 2 (PWM) of Arduino, which is connected to pin 6 of AVR CPU. Also, there is an LED connected to the Digital pin 40 of Arduino, which turns on once an authorized tag is scanned. Figure 3.16 shows all of the connections that were listed in this paragraph.

Figure 3.15 RFID Module Pins
Figure 3.16 RFID Module Connection to Arduino Scheme
3.8. Multi-Channel Relay

In this project, we have used a total of 12 relays (Figure 3.18) including an 8-Channel Relay and a 4-Channel Relay to control 12 different outputs which could control high-current appliances and equipment. For a better understanding of the concept, a single relay connection has shown in Figure 3.17. The 8-Channel relay board has 10 pins. VCC, GND and 8 INTs which are connected to the Arduino board. INT1 to INT8, which are the input pins of the relay board connected to the Arduino MEGA2560 as shown in the below table.

<table>
<thead>
<tr>
<th>INT pin on Relay</th>
<th>Output pin on Arduino</th>
<th>Pin# on AVR CPU</th>
<th>AVR CPU Pin Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>INT 7 of 8</td>
<td>Digital pin 33</td>
<td>57</td>
<td>PC4 ( A12 )</td>
</tr>
<tr>
<td>INT 6 of 8</td>
<td>Digital pin 35</td>
<td>55</td>
<td>PC2 ( A10 )</td>
</tr>
<tr>
<td>INT 5 of 8</td>
<td>Digital pin 37</td>
<td>53</td>
<td>PC0 ( A8 )</td>
</tr>
<tr>
<td>INT 4 of 8</td>
<td>Digital pin 39</td>
<td>70</td>
<td>PG2 ( ALE )</td>
</tr>
<tr>
<td>INT 3 of 8</td>
<td>Digital pin 41</td>
<td>51</td>
<td>PG0 ( WR )</td>
</tr>
<tr>
<td>INT 2 of 8</td>
<td>Digital pin 43</td>
<td>41</td>
<td>PL6</td>
</tr>
<tr>
<td>INT 1 of 8</td>
<td>Digital pin 45 (PWM)</td>
<td>39</td>
<td>PL4 ( OC5B )</td>
</tr>
</tbody>
</table>

The 4-Channel relay board has 6 pins. VCC, GND and 4 INTs which are connected to the Arduino board. INT1 to INT4 which are the input pins of the relay board connected to the Arduino MEGA2560 as it shows in the below table.

<table>
<thead>
<tr>
<th>INT pin on Relay</th>
<th>Output pin on Arduino</th>
<th>Pin# on AVR CPU</th>
<th>AVR CPU Pin Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>INT 1 of 4</td>
<td>Digital pin 42</td>
<td>42</td>
<td>PL7</td>
</tr>
<tr>
<td>INT 2 of 4</td>
<td>Digital pin 44 (PWM)</td>
<td>40</td>
<td>PL5 ( OC5C )</td>
</tr>
<tr>
<td>INT 3 of 4</td>
<td>Digital pin 46 (PWM)</td>
<td>38</td>
<td>PL3 ( OC5A )</td>
</tr>
<tr>
<td>INT 4 of 4</td>
<td>Digital pin 48</td>
<td>36</td>
<td>PL1 ( ICP5 )</td>
</tr>
</tbody>
</table>
Figure 3.17. A Single Relay Connections
Figure 3.18. An 8-Channel Relay Board and a 4-Channel Relay Board pin Mapping to an Arduino MEGA2560.
CHAPTER 4

SYSTEM SOFTWARE DESIGN

4.1. Introduction

This chapter describes the software system design, program techniques, and system approaches used in the development of programming the microprocessor in the smart home automation system. The software program is responsible for accepting data and commands, executing different commands, controlling operational terminals, and supporting data Input/output ports. This chapter discusses the software in terms of routines and subroutines. It is intended to give a general idea of program flow and implementation. All the routines’ and subroutines’ labels presented in this chapter are found in the C++ program listed in Appendix A. The listing of these routines and subroutines which are referenced in the flowcharts, can be found in appendix A. The reader should refer to Appendix A since it provides a detailed description about the program flow and implementation.

4.2. Program Description

The method used to design this software is a top-down structured software design, which divides the program into routines and subroutines (in another word, a modular program design). This method can help the programmer to test and debug the program easier, and can help the reader to read and understand it better. The program mainly consists of two routines. One is port addresses
and definitions routine, the other is the main routine. The flowchart is given in Figure 4.1. The following sections explain in detail the functions performed by routines and subroutines.

4.2.1. Port Addresses and Definitions

The program begins with port addresses and definitions routine, which defines all the port addresses, flags and pointers. It defines all input/output port addresses for all sensors, RFID, Ethernet shield, relays; also, time delay contours and control port addresses. Furthermore, it calls some C++ libraries and defines some initial values. The flowchart is shown in Figure 4.2.

4.2.2. Main Program

The main program routine (main), after the port addresses and definitions routines, consists of several subroutines. The flowchart for the main program routine is given in Figure 4.3. The program first calls different libraries, then the basic operation of the main routine is as follows:

4.2.2.1. Serial Peripheral Interface (SPI) Library

SPI “SPI.h” is a protocol that microcontroller uses to talk to one or multiple devices or another microcontroller. In this communication, there is the main device which mostly is a microcontroller, which controls the other devices. All of the devices have the below pins in common.

- SCK: (Serial Clock) are the clock pulses that coordinate data communication made by the main and single pin point to each device.
- MOSI: (Master Out Slave In) is the main pin for transferring data to the other devices.

- MISO: (Master In Slave Out) is the Slave pin for transferring data to the main.

4.2.2.2. Ethernet Library

By using Ethernet shield (W5100), the “Ethernet.h” library lets the microcontroller connect to the network. It could be used in the role of the server to receive, client to transfer, or both.

After calling this library, we need to initial MAC, Gateway, Subnet and IP address values and a specific server port to communicate over.

4.2.2.3. Temperature and Humidity Library

This library belongs to the DHT11 sensor and supports the function of reading the temperature and humidity sensor’s data easily with a “read ( )” command.

4.2.2.4. RFID Library

This library “rfid.h” includes the necessary information for all commands for communications between the microcontroller and the RFID reader, which uses Serial Peripheral Interface (SPI) for communications.
Figure 4.3 Flowchart of Main Routine
After calling all those libraries, it is time to define some initial specific pins and give them a specific value, after which there are some integers that need to be defined to get initial values.

4.3. Subroutines Within the Main Routine

The following sections discuss and explain the subroutines within the main routine.

4.3.1. Initialization Subroutine

This routine starts by initialization of all programmable ports to be either an Input or an Output and then each gets a LOW value as an initial value. pinMode( ) is the command to set them as an input or output. digitalWrite( ) is the command to give each pin an initial value. Flowchart of this subroutine is shown in Figure 4.4.

4.3.2. Check Device Flag Subroutine

This subroutine loads and checks the status of the device flag. The flowchart is shown in Figure 4.5.

4.3.3. Check Relay ON Subroutine

This subroutine sets the device relay “on” and it also sets the device flag. Figure 4.6 shows the flowchart.

4.3.4. Check Relay OFF Subroutine

This subroutine turns the device relay “off” and resets the device flag. Figure 4.7 shows the flowchart.
Figure 4.4 Flowchart for Initialization Subroutine

Figure 4.5 Flowchart for Check Device Flag Subroutine
Figure 4.6 Flowchart for Device Relay ON Subroutine
Figure 4.7 Flowchart for Device Relay OFF Subroutine
4.4. Subroutine Within the Network Interface

The following subroutines are within the Network interface subroutine (Figure 4.8)

“Ethernet.begin( )”

4.4.1. Server mode Subroutine “server.begin( )”

4.4.2. Ethernet Client mode Subroutine “EthernetClient ( )”

In this subroutine, all the programming for the client part will apply. Since in this project we have a web-based interface and a mobile App interface as a client part, we have used XML programming language to make a connection with the front end of the interface. Here we have programmed an interface to control 7 relays remotely via web browser and a smartphone application.

4.4.3. Time Out Flag Subroutine

This subroutine loads and checks the status of the timeout flag. The flowchart is given in Figure 4.9.

4.4.4. Time Delay Subroutine

This is a time delay subroutine. It uses delay( ) command to set the delay in milliseconds equal to the value in the parenthesis. Figure 4.10.
Figure 4.8 Flowchart for Network Interface Subroutine
Figure 4.9 Flowchart of Time Out Subroutine

Figure 4.10 Flowchart of Delay Subroutine
4.5. Photocell

In this part of the program, there is a loop including “if-else” that takes the data of the photocell sensor with “analogRead( )” command from the analog input pin of the Arduino, and then compares it with a defined initial value for darkness. If it is lower, then it sends a signal to an output pin of Arduino, which is connected to a relay, and turns the light on with “digitalWrite(RELAY_CH9, HIGH)” command. This loop always checks the value of the photocell and if the value of lightness goes high then it turns the light relay off with “digitalWrite(RELAY_CH9, LOW)” command as soon as the lighting value goes high (Flowchart 4.11).

4.6. Gas Sensor

Following to the photocell codes, programming for a Gas sensor starts. Normal gas pressure has a range that is defined in the initialization part of the program. This part of the loop has an “If-else” section which compares the taken value of the sensor via analog input pin of Arduino with the defined normal range. If the gas pressure is out of the normal range then it sends a signal to an output pin of the Arduino, which is connected to a relay and it turns the Fire Alarm on. The command for reading the gas pressure value is “analogRead( )” and the command for turning on the pin connected to the relay is “digitalWrite(RELAY_CH11, HIGH)” then if the gas pressure is normal it turns the Fire Alarm off with “digitalWrite(RELAY_CH11, LOW);” command. Figure 4.12 shows the flowchart for this part of the program.
Figure 4.11 Flowchart for Photocell Sensor

1. Read the Photocell Sensor Value
2. Check if it is DARK
   - Yes: Turn the Relay on
   - No: Turn the Relay off
3. Turn the Light off
Read the Gas Sensor Value

Gas Pressure is normal

Turn the Fire Alarm off

Turn the Fire Alarm on

Turn the Relay on
4.7. Temperature and Humidity

Following the Gas sensor codes, programming for a temperature and humidity sensor begins. Normal room temperature has a range that has been defined in the initialization part of the program. This part of the loop has an “If-else” section, which compares the taken value of the sensor via digital input pin of Arduino (pin 34) with a defined normal range. If the temperature is out of the normal range then it sends a signal to an output pin of the Arduino, which is connected to a relay and it turns the fan or AC on. The command for reading the temperature value is “DHT11.read()” and the command for turning on the pin connected to the relay is “digitalWrite(RELAY_CH12, HIGH)” then if the temperature is normal it turns the fan/AC off with “digitalWrite(RELAY_CH12, LOW);” command. Figure 4.13 shows the flowchart for this part of the program.

4.8. Motion Detection Sensor (PIR)

After the temperature and humidity sensor programs, the motion detection sensor programs start. This part of the program utilizes a nested loop that always keeps the system in the ready status in order to send a signal to a digital input pin of Arduino (pin 36) as soon as PIR’s infrared eye detects any movement in the area. Then Arduino sends a signal to an output pin (pin 31), which is connected to a relay and it turns the light on. The command for reading the motion detection sensor value is “digitalRead( )” and the command for turning the connected pin to relay on is “digitalWrite(RELAY_CH81, HIGH)” then if the sensor does not detect any movement, it turns the light off with “digitalWrite(RELAY_CH81, LOW);” command. Figure 4.14 shows the flowchart for this part of the program.
Figure 4.13 Flowchart for Temperature and Humidity Sensor
Figure 4.14 Flowchart for Motion Detection (PIR) Sensor
4.9. RFID Module

After the motion detection sensor codes, programming for the RFID module begins. There are tags, which are scannable by the RFID reader: each magnetic card ID (tag) has a unique code in hexadecimal. In this part of the program, there is a loop that is always checking the RFID module and as soon as any tag gets scanned, the program will compare the scanned tag code (ID) with all defined existing tag codes in the program. If the code matches, it is an authorized tag holder and will gain access (to the gate/door).

Once a tag gets scanned then RFID module sends the data (including the tag code) to the Arduino digital input pin via six lines (See section 3.7 for detailed pin mapping information) with “rfid.request( )” command, the program compares the tag’s code and if it matches, then it sends a signal to a digital output pin (pin 44) of Arduino with “digitalWrite(RELAY_CH10, 1)” command, which is connected to the relay int2 of 4-chanel relay. At the same time it turns the LED on with “digitalWrite(rfPin, 1)” command, then it has a delay for 2000ms and then it sets the relay and the LED off via “digitalWrite(rfPin, 0)” and “digitalWrite(RELAY_CH10, 0)” commands which lock the door/gate again. Figure 4.15 shows the flowchart for this part of the program.
Read the Tag/Card ID code

Authorized Tag/Card Holder

Yes

Turn the Relay on

Unlock the door

Delay 2000 ms

Turn off the Relay

Door is locked

No

Keep the Relay off

Door stays locked

Figure 4.15 Flowchart for RFID Module
4.10. Android Application Development

This is an open source Android application which has a graphical interface. The Android application codes have been written with Android SDK which is offered by Android developer’s blog. It is a GUI (Graphical User Interface) which makes the job for building an application much easier.

The interface only has a graphic designed part and a setting for the network connection in the back end. Besides all those graphic designs, the network connection settings are the main part, which need to be set to the defined IP address and port number that originally were initialized in the Arduino program initialization section (see section 4.2.2.2 for). When the network connection is set, the application communicates with the main Arduino program (Figure 4.16). The main program has an XML code below the server part of the network interface (See section 4.4). The main Arduino program makes a connection between the network interface as a server and the Android application as a client, and it sends and receives the ON/OFF commands from the Android application interface. All those commands get possessed in the main Arduino program, which sends the signal to Digital output pins 45,43,41,39,37,35,33 to Int2 to Int8 of the 8-Channel relay to ON/OFF each relay (Figure 4.17).
Figure 4.16 Flowchart for an Android Application Interaction with Arduino Program
Figure 4.17 Flowchart for the Main Program Interaction with a Remote User
CHAPTER 5

CONCLUSION

5.1. System Evaluation

In this project, a unique Smart Home Automation System has been designed. A prototype has been built, and its performance has been tested and has been proven to work as expected.

There were several problems that we encountered during the hardware troubleshooting.

i) There was no action among Input/output ports. That is, when data was sent to I/O ports, nothing appeared on the I/O output port pins. This problem was caused due to the lack of experience of wiring between sensor modules and the Arduino microcontroller. Searching on the internet and watching online tutorials helped me in finding out differences between analog and digital ports of the Arduino and how to map those ports in the programming part of my project; then this issue got resolved by remapping ports/pins in both hardware and software parts.

ii) There was a problem with the network connection between the network interface and the remote user. The system was not able to communicate with the remote user. It was because of the lack of information in Client/Server communication with the microprocessor in both client and server site of the program. The problem got solved by matching the IP address and port number in both client and server site.
Another problem was the relays’ status showing on the cell phone application which needed to get updated in real-time. Since other modules needed to have a delay, it was not able to work on time. Changing the sequence in the loop part of the main program helped this issue get resolved.

5.2. Conclusion

The computer board circuit performs very well. All sensors send data to the microcontroller in real-time. The RFID module performs immediately and gives access with enough delay time for the user to open the actual door/gate. Network connection is completely established and the mobile application could control relays in real-time.

This project can be expanded to different uses. Moreover, many other modules and sensors can be added.

In this project, I have utilized a wired Ethernet shield for communication between the microcontroller and home area network (HAN); the same project could use a wireless Ethernet module instead. Furthermore, we can have some additional systems like this one in different zones of the house, which are capable of communicating with each other wirelessly via WLAN or newer short-range wireless technologies such as Bluetooth and ZigBee (XBee).
REFERENCES:


APPENDIX A

ARDUINO MICROCONROLLER PROGRAM LIST

#include <SPI.h> // calling Serial Peripheral Interface (SPI) Library

#include <Ethernet.h> // calling temperature ethernet shield library

#include <dht11.h> // calling temperature sensor library

#include "rfid.h" // calling RFID sensor library

#define mq2Pin 1  // MQ-2 module attach to A1 on MEGA2560

#define mqMax 700 // gas pressure maximum number

#define DHT11PIN 34  // DHT11 attach to digital pin 34 on controller board

#define photocellPin 0  // photocell attach to A0 on MEGA2560

#define cellMax 800 // photocell maximum number

#define cellMin 200 // photocell minimum number

DHT11 DHT11;

RFID rfid;

uchar serNum[5];

// Enter a MAC address and IP address for your controller below.

// The IP address will be dependent on your local network:
byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0x2D }; //MAC address of the controller

byte gateway[] = { 192, 168, 0, 1 }; //<------ ROUTERS IP Address to

byte subnet[] = { 255, 255, 255, 0 }; //<------ Router subnet address

IPAddress ip(192,168,0,199); //set the IP address

EthernetServer server(80); //port 80 for HTTP (transferring port)

#define RELAY_CH1 45 //pin 45 of Arduino defined as a RELAY_CH1
#define RELAY_CH2 43 //``43`````````````````````````````````````````````````````````2
#define RELAY_CH3 41 //``41`````````````````````````````````````````````````````````3
#define RELAY_CH4 39 //````39`````````````````````````````````````````````````````````4
#define RELAY_CH5 37 //``````37`````````````````````````````````````````````````````````5
#define RELAY_CH6 35 //````````35`````````````````````````````````````````````````````````6
#define RELAY_CH7 33 //``````````33```````````````````````````````````````````````````````7
#define RELAY_CH8 30 //````````````30`````````````````````````````````````````````````````8
#define RELAY_CH81 31 //``````````````31`````````````````````````````````````````````````9

#define pirPin 36 //PIR attach to pin 36
#define ledPin 38 //LED attach to pin 38
#define RELAY_CH9 42 //rfid relay attach to pin42
#define rfPin 40 //rfid LED attach to pin 40
```
#define RELAY_CH10 44  // LED attach to pin 44
#define RELAY_CH11 46 //Gas presure to relay int3 =>Fire Alarm
#define RELAY_CH12 48 //temp => AC

#define uint8 unsigned char //initialize character
#define uint16 unsigned int // initialize unsigned character
#define uint32 unsigned long // initialize unsigned character

int pirValue = 0; //initialization – set PIR value to 0
int pirState = LOW; // initialization – set PIR status to OFF

String readString; //define a string

void setup() {

  pinMode(RELAY_CH1, OUTPUT); // Arduino’s pin defines RELAY_CH1 as output
  digitalWrite(RELAY_CH1, LOW);  // LOW initial value to RELAY_CH1

  pinMode(RELAY_CH2, OUTPUT); // Arduino’s pin defines RELAY_CH2 as output
  digitalWrite(RELAY_CH2, LOW);  // LOW initial value to RELAY_CH2

  pinMode(RELAY_CH3, OUTPUT); // Arduino’s pin defines RELAY_CH3 as output
  digitalWrite(RELAY_CH3, LOW);  // LOW initial value to RELAY_CH3

  pinMode(RELAY_CH4, OUTPUT); // Arduino’s pin defines RELAY_CH4 as output
  digitalWrite(RELAY_CH4, LOW);  // LOW initial value to RELAY_CH4

  pinMode(RELAY_CH5, OUTPUT); // Arduino’s pin defines RELAY_CH4 as output

```
digitalWrite(RELAY_CH5, LOW); // LOW initial value to RELAY_CH5

pinMode(RELAY_CH6, OUTPUT); // Arduino’s pin defines RELAY_CH6 as output

digitalWrite(RELAY_CH6, LOW); // LOW initial value to RELAY_CH6

pinMode(RELAY_CH7, OUTPUT); // Arduino’s pin defines RELAY_CH7 as output

digitalWrite(RELAY_CH7, LOW); // LOW initial value to RELAY_CH7

pinMode(RELAY_CH8, OUTPUT); // Arduino’s pin defines RELAY_CH8 as output

digitalWrite(RELAY_CH8, LOW); // LOW initial value to RELAY_CH8

// Start serial communications and hold until port gets leased:

Serial.begin(9600);

while (!Serial) {
    ;
}

// beginning of the network connection and the server:

Ethernet.begin(mac, ip); // checks if the MAC and IP are matched

server.begin(); // set to work as server

Serial.print("server is at "); // print on LCD

Serial.println(Ethernet.localIP()); // print on LCD

rfid.begin(2, 3, 4, 5, 6, 7); // rfid connected to those digital pin of Arduino
delay(100); //delay

rfid.init(); //reading the rfid library

pinMode(pirPin, INPUT); // Arduino’s pin defines pirPin as Input

pinMode(ledPin, OUTPUT); // Arduino’s pin defines ledPin as output

pinMode(RELAY_CH81, OUTPUT); // Arduino’s pin defines RELAY_CH81 as output

digitalWrite(RELAY_CH81, LOW); // LOW initial value to RELAY_CH81

pinMode(RELAY_CH9, OUTPUT); // Arduino’s pin defines RELAY_CH9 as output

digitalWrite(RELAY_CH9, LOW); // LOW initial value to RELAY_CH9

pinMode(RELAY_CH10, OUTPUT); // Arduino’s pin defines RELAY_CH10 as output

pinMode(RELAY_CH11, OUTPUT); // Arduino’s pin defines RELAY_CH11 as output

digitalWrite(RELAY_CH11, LOW); // LOW initial value to RELAY_CH11

pinMode(RELAY_CH12, OUTPUT); // Arduino’s pin defines RELAY_CH12 as output

digitalWrite(RELAY_CH12, LOW); // LOW initial value to RELAY_CH12

}

void loop() {  
  EthernetClient client = server.available();  // listening for incoming clients

  if (client) {

    Serial.println("new client");  // print on dispaly

    boolean currentLineIsBlank = true; //checks if its blank it terminated the connection

  }
}
while (client.connected()) { // which connection is established do the below commands

if (client.available()) { // check if the client is available

char c = client.read() // assign the read value to c which is a “character”

if (readString.length() < 100) { // check how long is the character string

readString += c; // reading next string

Serial.print(c); // print the value on the screen
}

Serial.write(c);

if (c == 'n' && currentLineIsBlank) {

// refer a typical http reply header

client.println("HTTP/1.1 200 OK"); // checks if the connection is established then print on display

client.println("Content-Type: text/html"); // print on display

client.println("Connection: close"); // the communication will be terminated once response is completed.

client.println("Refresh: 5"); // re-fresh the connection routinely each five second

client.println(); // print on display

client.println("<!DOCTYPE HTML>"); // html coding header command

client.println("<html>"); // html coding header command

client.println("<HEAD>"); // html coding header command

client.println("<meta name='apple-mobile-web-app-capable' content='yes' />");

client.println("<meta name='apple-mobile-web-app-status-bar-style' content='black-translucent' />");

client.println("</HEAD>"); //start programming XML for web based app

client.println("<body bgcolor="#D0D0D0">"); //Background color

src="https://lh3.googleusercontent.com/-C6BoJrRUFko/UEUFecwkvdI/AAAAAAAAAOc/E7gcYvPV6r4/s960/Logo.jpg"

</center></h4>);

client.println("<center> <p> <h1>Welcome to Arduino Home Control V0.25 </h1></p> "); //Display message in web browser

client.println("<center>"); //start tables

client.println("<table border="5">"); //set table border

client.println("<tr>"); //table command

if (digitalRead(RELAY_CH1)) // check the relay in the web browser app
{

client.print("<td> <p style="font-family:arial;color:black;font-size:26px;"">Device 1.</p><p style="font-family:arial;color:green;font-size:35px;"">ON</p> </td> ");

} // XML styling for buttons belong to relay 1 when it is ON

else {


client.print("<td> <p style="font-family:arial;color:black;font-size:26px;">Device 1.</p><p style="font-family:arial;color:red;font-size:35px;">OFF</p></td>");

} // XML styling for buttons belong to relay 1 when it is OFF

if (digitalRead(RELAY_CH2))
{
  client.print("<td> <p style="font-family:arial;color:black;font-size:26px;">Device 2.</p><p style="font-family:arial;color:green;font-size:35px;">ON</p></td>");
}

} // XML styling for buttons belong to relay 2 when it is ON
else
{
  client.print("<td> <p style="font-family:arial;color:black;font-size:26px;">Device 2.</p><p style="font-family:arial;color:red;font-size:35px;">OFF</p></td>"); // XML styling for buttons belong to relay 2 when it is OFF
}

if (digitalRead(RELAY_CH3))
{
  client.print("<td> <p style="font-family:arial;color:black;font-size:26px;">Device 3.</p><p style="font-family:arial;color:green;font-size:35px;">ON</p></td>");

  // XML styling for buttons belong to relay 3 when it is ON
}
else
{
  client.print("<td> <p style="font-family:arial;color:black;font-size:26px;">Device 3.</p><p style="font-family:arial;color:red;font-size:35px;">OFF</p></td>"); // XML styling for buttons belong to relay 3 when it is OFF
}

if (digitalRead(RELAY_CH4))

{ client.print("<td><p style="font-family:arial;color:black;font-size:26px;">Device 4. </p><p style="font-family:arial;color:green;font-size:35px;">ON</p></td>");

    // XML styling for buttons belong to relay 4 when it is ON  }

else { client.print("<td><p style="font-family:arial;color:black;font-size:26px;">Device 4. </p><p style="font-family:arial;color:red;font-size:35px;">OFF</p></td>");

    // XML styling for buttons belong to relay 4 when it is OFF  }

if (digitalRead(RELAY_CH5))
{
    client.print("<td><p style="font-family:arial;color:black;font-size:26px;">Device 5. </p><p style="font-family:arial;color:green;font-size:35px;">ON</p></td>");

    // XML styling for buttons belong to relay 5 when it is ON  }

else
{

    client.print("<td><p style="font-family:arial;color:black;font-size:26px;">Device 5. </p><p style="font-family:arial;color:red;font-size:35px;">OFF</p></td>");

    // XML styling for buttons belong to relay 5 when it is OFF  }

if (digitalRead(RELAY_CH6))
// XML styling for buttons belong to relay 6 when it is ON

} else {
  client.print("<td><p style="font-family:arial;color:black;font-size:26px;">Device 6.<p style="font-family:arial;color:green;font-size:35px;">ON</p></td>" once the relay is turned OFF in the code.

// XML styling for buttons belong to relay 6 when it is OFF

if (digitalRead(RELAY_CH7))

{ client.print("<td><p style="font-family:arial;color:black;font-size:26px;">Device 7.<p style="font-family:arial;color:green;font-size:35px;">ON</p></td>" once the relay is turned OFF in the code.

// XML styling for buttons belong to relay 7 when it is ON

} else

{ client.print("<td><p style="font-family:arial;color:black;font-size:26px;">Device 7.<p style="font-family:arial;color:red;font-size:35px;">OFF</p></td>" once the relay is turned OFF in the code.

// XML styling for buttons belong to relay 7 when it is OFF

if (digitalRead(RELAY_CH8))

{ client.print("<td><p style="font-family:arial;color:black;font-size:26px;">Device 8.<p style="font-family:arial;color:green;font-size:35px;">ON</p></td>" once the relay is turned OFF in the code.
// XML styling for buttons belong to relay 8 when it is ON
}

else { client.print("<td><p style="font-family:arial;color:black;font-size:26px;">Device 8.<p style="font-family:arial;color:red;font-size:35px;">OFF</p></td>");

// XML styling for buttons belong to relay 8 when it is OFF
}

client.println("</tr>");

client.println("</table>");

client.println("</center>");

client.println("<br />");

client.println("<a href="/?relay1on" > <button style="width:360px;height:120px"> <font size="7"; color:red ;>Device 1 ON </font> </button> </a> ");

client.println("<a href="/?relay1off" > <button style="width:360px;height:120px"> <font size="7">Device 1 OFF </font> </button> </a> <br />");

client.println("<br />");

client.println("<a href="/?relay2on" > <button style="width:360px;height:120px"> <font size="7"; color:red ;>Device 2 ON </font> </button> </a> ");

client.println("<a href="/?relay2off" > <button style="width:360px;height:120px"> <font size="7">Device 2 OFF </font> </button> </a> <br />");

client.println("<br />");
client.println("<a href="/?relay3on" > <button style="width:360px;height:120px" > <font size="7">Device 3 ON </font> </button> </a> ");

client.println("<a href="/?relay3off" > <button style="width:360px;height:120px" > <font size="7">Device 3 OFF </font> </button> </a> <br />");

client.println("<br />");

client.println("<a href="/?relay4on" > <button style="width:360px;height:120px" > <font size="7">Device 4 ON </font> </button> </a> ");

client.println("<a href="/?relay4off" > <button style="width:360px;height:120px" > <font size="7">Device 4 OFF </font> </button> </a> <br />");

client.println("<br />");

client.println("<a href="/?relay5on" > <button style="width:360px;height:120px" > <font size="7">Device 5 ON </font> </button> </a> ");

client.println("<a href="/?relay5off" > <button style="width:360px;height:120px" > <font size="7">Device 5 OFF </font> </button> </a> <br />");

client.println("<br />");

client.println("<a href="/?relay6on" > <button style="width:360px;height:120px" > <font size="7">Device 6 ON </font> </button> </a> ");

client.println("<a href="/?relay6off" > <button style="width:360px;height:120px" > <font size="7">Device 6 OFF </font> </button> </a> <br />");

client.println("<br />");
client.println("<a href="/?relay7on">Device 7 ON <button style="width:360px;height:120px"> <font size="7">Device 7 ON</font> </button> </a>");

client.println("<a href="/?relay7off">Device 7 OFF <button style="width:360px;height:120px"> <font size="7">Device 7 OFF</font> </button> </a> <br />");

client.println("<br />");

client.println("<a href="/?relay8on">Device 8 ON <button style="width:360px;height:120px"> <font size="7">Device 8 ON</font> </button> </a> ");

client.println("<a href="/?relay8off">Device 8 OFF <button style="width:360px;height:120px"> <font size="7">Device 8 OFF</font> </button> </a> <br />");

client.println("<br />");

/* ----- from here control Arduino pins by Networks shield begins------- */

if(readString.indexOf("?relay1on") >0)// checks relay for ON status
{

digitalWrite(RELAY_CH1, HIGH); // set pin relay ch1’s pin to High

client.println("<link rel='apple-touch-icon' href='http://chriscosma.co.cc/on.png' />");

client.println("<br />");
}

else{

if(readString.indexOf("?relay1off") >0)// checks relay for OFF status

digitalWrite(RELAY_CH1, LOW);  // set pin relay ch1’s pin to LOW

client.println("<link rel='apple-touch-icon' href='http://chriscosma.co.cc/off.png' />");

client.println("<br />");

} 

} 

if(readString.indexOf("?relay2on") >0) // checks relay for ON status

{

digitalWrite(RELAY_CH2, HIGH);  // set pin relay ch2’s pin to High

client.println("<link rel='apple-touch-icon' href='http://chriscosma.co.cc/on.png' />");

client.println("<br />");

} 

else{

if(readString.indexOf("?relay2off") >0) // checks relay for OFF status

{

digitalWrite(RELAY_CH2, LOW);  // set pin relay ch2’s pin to LOW

client.println("<link rel='apple-touch-icon' href='http://chriscosma.co.cc/off.png' />");

client.println("<br />");

}
if(readString.indexOf("?relay3on") >0) // checks relay for ON status
{
  digitalWrite(RELAY_CH3, HIGH); // set pin relay ch3’s pin to HIGH
  client.println("<link rel='apple-touch-icon' href='http://chriscosma.co.cc/on.png' />");
  client.println("<br />");
}
else{
  if(readString.indexOf("?relay3off") >0) // checks relay for OFF status
  {
    digitalWrite(RELAY_CH3, LOW); // set pin relay ch3’s pin to LOW
    client.println("<link rel='apple-touch-icon' href='http://chriscosma.co.cc/off.png' />");
    client.println("<br />");
  }
}

if(readString.indexOf("?relay4on") >0) // checks relay for ON status
{
  digitalWrite(RELAY_CH4, HIGH); // set pin relay ch4’s pin to HIGH
  client.println("<link rel='apple-touch-icon' href='http://chriscosma.co.cc/on.png' />");
client.println("<br />");

} else{

    if(readString.indexOf("?relay4off") >0) // checks relay for OFF status

    {

digitalWrite(RELAY_CH4, LOW);  // set pin relay ch4’s pin to LOW

client.println("<link rel='apple-touch-icon' href='http://chriscosma.co.cc/off.png' />");

client.println("<br />");

    }

} 

}

if(readString.indexOf("?relay5on") >0) // checks relay for ON status

{

digitalWrite(RELAY_CH5, HIGH);  // set pin relay ch5’s pin to HIGH

client.println("<link rel='apple-touch-icon' href='http://chriscosma.co.cc/on.png' />");

client.println("<br />");

}

else{

    if(readString.indexOf("?relay5off") >0) // checks relay for OFF status
```c
{
    digitalWrite(RELAY_CH5, LOW);  // set pin relay ch5’s pin to LOW

    client.println("<link rel='apple-touch-icon' href='http://chriscosma.co.cc/off.png' />");

    client.println("<br />");
}

if(readString.indexOf("?relay6on") >0) // checks relay for ON status
{
    digitalWrite(RELAY_CH6, HIGH);  // set pin relay ch6’s pin to HIGH

    client.println("<link rel='apple-touch-icon' href='http://chriscosma.co.cc/on.png' />");

    client.println("<br />");
}
else{
    if(readString.indexOf("?relay6off") >0) // checks relay for OFF status
    {
        digitalWrite(RELAY_CH6, LOW);  // set pin relay ch6’s pin to LOW

        client.println("<link rel='apple-touch-icon' href='http://chriscosma.co.cc/off.png' />");

        client.println("<br />");
    }
```
if(readString.indexOf("?relay7on") > 0) // checks relay for ON status
{
    digitalWrite(RELAY_CH7, HIGH); // set pin relay ch7’s pin to HIGH
    client.println("<link rel='apple-touch-icon' href='http://chriscosma.co.cc/on.png' />");
    client.println("<br />");
}

else{
    if(readString.indexOf("?relay7off") > 0) // checks relay for OFF status
    {
        digitalWrite(RELAY_CH7, LOW); // set pin relay ch7’s pin to LOW
        client.println("<link rel='apple-touch-icon' href='http://chriscosma.co.cc/off.png' />");
        client.println("<br />");
    }
}

if(readString.indexOf("?relay8on") > 0) // checks relay 8 for ON status
{

digitalWrite(RELAY_CH8, HIGH);  // set pin relay ch8’s pin to HIGH

client.println("<link rel='apple-touch-icon' href='http://chriscosma.co.cc/on.png' />");

client.println("<br />");

}

else{

    if(readString.indexOf("?relay8off") >0) // checks relay 8 for OFF status

    {

        digitalWrite(RELAY_CH8, LOW);  // set pin relay ch8’s pin to LOW

        client.println("<link rel='apple-touch-icon' href='http://chriscosma.co.cc/off.png' />");

        client.println("<br />");

    }

}

}

}

client.println("<hr> <p> By <a href="www.linkedin.com/pub/ehsan-dolatshahi/9a/321/280/"&gt;&lt;/a&gt;"">EHSAN DOLATSHAHI</p>");

readString="";

client.println("</body>");

client.println("</html>");
break;
}
if (c == '\n') {
    currentLineIsBlank = true;
}
else if (c != '\r') {
    currentLineIsBlank = false;
}
delay(1); // provide some time to web browser to get the data.

client.stop(); // suspend the connection
Serial.println("client disconnected"); // disconnect the connection

// ----------------------------------------------------------- PHOTO CELL ---------------

int num;
Serial.print("Photocell Value: "); // write photocell value on display
num = analogRead(photocellPin); // reading analog pin belong to photocell
if(num <= cellMin){

    // compare the photocell value to the defined value to minimum lightness

digitalWrite(RELAY_CH9, HIGH); // it’s dark turn on the relay9

} //turn on lights

else{

digitalWrite(RELAY_CH9, LOW); // it is light enough, turn off the light.

}

Serial.println(num);

//------------------------------------------END OF PHOTOCELL

//------------------------------------------MQ-2 – GAS SENSOR

int num2;

Serial.print("MQ-2 Value: "); //print gas pressure value on the display

num2 = analogRead(mq2Pin); //read value of the gas pressure from Arduino analog pin

if(num2 >= mqMax){

    // compare the value of gas pressure to the normal gas pressure rage

    digitalWrite(RELAY_CH11, HIGH);

    /* if the value is not in the normal range, turn on the relay11 (fire alarm) */

}else{
//if(num2 < mqMax){

digitalWrite(RELAY_CH11, LOW);

/* if the value is in the normal range, turn off the relay11 (fire alarm) */
}

Serial.println(num2);

//-------------------------------------END OF MQ-2

//-------------------------------------TEMP SENSOR

Serial.print("DHT11,\t");

int chk = DHT11.read(DHT11PIN); //read the value returned from temperature sensor

switch (chk) //switch case for sensor availability status
{

case DHTLIB_OK:

    Serial.print("OK,\t");

    break;

case DHTLIB_ERROR_CHECKSUM:

    Serial.print("Checksum error,\t");

    break;

case DHTLIB_ERROR_TIMEOUT:


Serial.print("Time out error,\n");
break;
default:
Serial.print("Unknown error,\n");
break;
}
Serial.println(); // print DATA information on the display
Serial.print("Tem: ");
Serial.print(DHT11.temperature); //display temp value on serial monitor
if(DHT11.temperature >= 29)
//check if the temperature is in the normal room temperature
{

digitalWrite(RELAY_CH12, HIGH);
/* if the value is not in the normal range, turn on the relay12 (fan) */
}else{

digitalWrite(RELAY_CH12, LOW);
/* if the value is in the normal range, turn off the relay12 (fan) */
} // display temperature information on serial display
Serial.println(" C");

Serial.print("Hum: ");

Serial.print(DHT11.humidity); //print the humidity on serial monitor

Serial.println(" %");

Serial.println();

//-------------------------------END OF TEMP

//-------------------------------PIR motion detection

pirValue = digitalRead(pirPin);

// read value of Arduino pin which is connected to the sensor

if (pirValue == HIGH){ // check if there is a motion? = yes

digitalWrite(ledPin, pirValue); // turn on the LED belong to motion detection

digitalWrite(RELAY_CH81, pirValue);

// turn on the relay 81 (lamp) belong to motion detection

if (pirState == LOW){ // check the status of the PIR

    pirState = HIGH; // initial value in the local loop

}

}else{

digitalWrite(ledPin, LOW); // turn off the LED belong to PIR
digitalWrite(RELAY_CH81, pirValue); //turn off relay 81 belong to PIR

if(pirState == HIGH){ //check the status of the PIR
    pirState = LOW; //initial value in the local loop
}

//--------------------------------------
END OF--PIR motion detection

//--------------------------------------RFID

uchar status;
uchar str[MAX_LEN];

status = rfid.request(PICC_REQIDL, str); // read the rfid

if (status != MI_OK) // check the status of rfid
{
    return;
}

rfid.showCardType(str);

status = rfid.anticoll(str);

if (status == MI_OK)
Serial.print("The card's number is: ");

// display scanned card number on the serial monitor

memcpy(serNum, str, 5);

rfid.showCardID(serNum);

Serial.println();

// Check if the scanned tag is authorized

uchar* id = serNum;

if( id[0]==0x54 && id[1]==0x26 && id[2]==0xBC && id[3]==0x96 ) {

    // check the code of tag

    Serial.println("Hello Ehsan!"); // if its match display user name of serial monitor

    digitalWrite(rfPin, 1); // rfid led turns on

    digitalWrite(RELAY_CH10, 1); // rfid relay on

    delay(2000); // wait time

    digitalWrite(rfPin, 0); // rfid led off anter 2000 ms

    digitalWrite(RELAY_CH10, 0); // rfid rely off after 2000 ms

}

else if(id[0]==0x15 && id[1]==0x6F && id[2]==0x9F && id[3]==0xAF)
{

Serial.println("Collect!");

Serial.println();

}

else
{

Serial.println("Hello unlown guy!");

// if the user is not authorized it show this message on the serial monitor

}

}

} //Loop

//-------------------------------END OF RFID
APPENDIX B

ANDROID PROGRAM SETTING CODES

<?xml version="1.0" encoding="utf-8"?>

<PreferenceScreen

xmlns:Android="http://schemas.android.com/apk/res/Android">

<PreferenceCategory Android:title="Setting">

<PreferenceScreen Android:title="@string/menu_set_button"

Android:key="setButtonName" Android:summary="">

<intent Android:action="arduino.homecontrol.homeautomation.Set_button_name" />

</PreferenceScreen>

<PreferenceScreen Android:title="@string/menu_set_sensor"

Android:key="setSensorName" Android:summary="">

<intent Android:action="arduino.homecontrol.homeautomation.Set_sensor_name" />

</PreferenceScreen>

<EditTextPreference Android:name="@id/menu_set_ip"


Android:defaultValue="192.168.0.199" />
<EditTextPreference Android:name="@id/menu_set_ip"
Android:defaultValue="80" Android:inputType="number" />

<PreferenceScreen Android:title="@string/menu_set_default" Android:key="setDefault"
Android:summary=""/>

<intent Android:action="arduino.homecontrol.homeautomation.ClearPage" />

</PreferenceScreen>

</PreferenceCategory>

</PreferenceScreen>
APPENDIX C

DATASHEETS

Atmel ATmega 2560 datasheet is available in the below link.


Ethernet Shield w5100 datasheet is available in the link below.


RFID RC522 datasheet is available in the below link.


PIR (motion detection) sensor datasheet is available in the below link.


Temperature and Humidity sensor (DHT11) datasheet is available in the below link.


Gas pressure sensor (MQ-2) datasheet is available in the below link.


Photocell (light sensor) datasheet is available in the below link.


Multi-channel relay board datasheet is available in the link below.