API's library for Arduino board Simulator using Javascript

A Project Report to be submitted in partial fulfillment of the requirements for the degree

of

Master Of Computer Application

by

Karan Soren
17MCMC12

SCHOOL OF COMPUTER AND INFORMATION SCIENCES
UNIVERSITY OF HYDERABAD, GACHIBOWLI
HYDERABAD - 500046,INDIA
JULY,2021

CERTIFICATE

This is to certify that we have studied the thesis entitled **API's library for Arduino board Simulator using Javascript**, submitted by **Karan Soren**(Roll Number: *17MCMC12*) a postgraduate student of **Department of School of Computer and Information Sciences** in partial fulfillment for the award of degree of Master Of Computer Application. We hereby accord our approval of it as a study carried out and presented in a manner required for its approval in partial fulfillment for the Post Graduate Degree for which it has been submitted. The project has fulfilled all the requirements as per the regulations of the Department and has reached the standard needed for submission.

Supervisor : Dr. Nagendra Kumar Suryadevara,
School of Computer and Information Sciences,
University of Hyderabad,Hyderabad

Dean,
School of Computer and Information Sciences,
University of Hyderabad,Hyderabad

Date: July 16, 2021

ACKNOWLEDGEMENTS

It is a great pleasure to thank the giants on whose shoulders I stand. First of all, I would like to express my deep gratitude to all those who helped and guided me throughout the project. I am deeply indebted to my guidance Nagendra Sir for giving me the opportunity to work on this challenging project and for being the pillar of strength for me throughout my project lifecycle during this pandemic. I also wish to express my gratitude to him for his guidance,valuable suggestions and for sharing his vast reservoir of knowledge and experience during the entire duration of project.My stay in the Department of School of Computer and Information Sciences has been a great learning experience and a curtain raiser to an interesting and rewarding career.I want to thank my teachers for having faith in me. And most importantly my parents to support me in all odds and moral support..

Karan Soren
University of Hyderabad,
Hyderabad
Date: July 16, 2021

Contents

1 Introduction **1**

1.1 About Arduino 1

1.1.1 Arduino Uno and its components 1

1.2 Need for an Arduino Simulator 3

1.3 Existing Arduino Simulator 4

1.4 Advantages of Web-Based Simulators 4

1.5 Need for an Application Programming Interface (API) for Arduino Simulators 5

1.6 Aim and objective of the projects 5

1.7 Organization of the thesis 6

2 User Requirements **8**

2.1 Introduction 8

2.2 Functional Requirements 9

2.3 Non-Functional Requirements 10

2.4 Designing Considerations 10

2.5 Software Technologies 11

3 Working of Arduino Board **12**

3.1 Introduction 12

3.2 The Components of the Arduino 12

3.2.1 Resistors 12

3.3 The Output Components 13

3.3.1 LEDs 13

3.3.2 Fan motor 13

3.3.3 Servo Motor 13

3.3.4 LCD . . . 14
3.4 The Input Components . . . 14
3.4.1 TH02 . . . 14
3.4.2 Push Button . . . 15
3.4.3 Matrix Keypad . . . 16
3.5 Arduino connecting to an API . . . 16
3.6 Installation . . . 18
3.6.1 Johnny-five . . . 20
3.7 Summary of the Chapter . . . 22

4 Proteus Design Suite **23**
4.1 Schematic Design using Proteus . . . 23
4.1.1 Master board and Slave board 1 . . . 24
4.1.2 Master board and Slave board 2 . . . 25

5 Implementation and Code **27**
5.1 API Methods . . . 27
5.1.1 com0com . . . 27
5.1.2 Arduino Standard Firmata . . . 28
5.1.3 Node js . . . 30
5.1.4 Visual Studio Code . . . 32

6 Execution and Results **47**
6.1 Unit Testing . . . 47
6.2 Integration Testing . . . 48
6.3 Results . . . 49

7 Task Analysis **52**

8 Conclusion and Future Work **54**
8.1 Conclusion . . . 54
8.2 Future Work . . . 54

Bibliography **56**

List of Figures

1.1 Arduino Uno 2

3.1 Two Resistors with 220 ohm and other 250 ohm 12
3.2 Arduino is connected with a led and resistor with 100 ohms 13
3.3 Arduino is connected with a fan motor and resistor with 100 ohms . . 14
3.4 Arduino is connected with a servo motor and resistor with 220 ohms . 14
3.5 LCD connected with arduino 15
3.6 One TH02 components with LCD displaying relative humidity 15
3.7 Pushbutton to turn On or Off the Fan 16
3.8 Matrix keypad connected to Arduino 16
3.9 API between Client and System 17
3.10 API between Arduino and System 18
3.11 installing johnny-five 18
3.12 installing socket io 19
3.13 client sending and requesting data from Server 19
3.14 Installing express 20
3.15 Led on and off using Johnny-Five 21

4.1 Schematic design 24

5.1 Creating a virtual port for the Arduino simulator and computer system 27
5.2 command for setting virtual port COM4 28
5.3 command for setting virtual port COM3 28
5.4 Getting Standard Firmata from Arduino IDE 29
5.5 Downloading Arduino IDE 30
5.6 Downloading Nodes Js for Windows 31
5.7 Command for verifying node and npm version 32
5.8 Downloading Visual Studio Code for Windows 32

5.9 Johnny-five Code implementing from line 1 to line line 19 33
5.10 Johnny-five Code implementing from line 21 to line 47. 34
5.11 Johnny-five Code implementing from line 48 to line line 77 34
5.12 Johnny-five Code implementing from line 71 to line 86 35
5.13 Johnny-five Code implementing from line 87 to line 90 35
5.14 Johnny-five Code implementing from line 92 to line 96 35
5.15 Johnny-five Code implementing from line 99 to line 103 36
5.16 Johnny-five Code implementing from line 142 to line 153 36
5.17 Johnny-five Code implementing from line 156 to line 180 36
5.18 Johnny-five Code implementing from line 182 to line 193 37
5.19 Johnny-five Code implementing from line 197 to line 215 37
5.20 Johnny-five Code implementing from line 281 to line 300 38
5.21 Johnny-five Code implementing from line 302 to line 326 38
5.22 Slave board 1 code of first part using Arduino IDE 39
5.23 Slave board 1 code of second part using Arduino IDE 40
5.24 Slave board 1 code of third part using Arduino IDE 41
5.25 Slave board 1 code of fourth part using Arduino IDE 41
5.26 Slave board 2 code of first part using Arduino IDE 42
5.27 Slave board 2 code of second part using Arduino IDE 43
5.28 Slave board 2 code of third part using Arduino IDE 44
5.29 Slave board 2 code of fourth part using Arduino IDE 45
5.30 Slave board 2 code of fifth part using Arduino IDE 46

6.1 Webpage that successfully controls Arduino board using js sever node 50
6.2 After completing all tests, the simulation was successful 51

7.1 Use case diagram 53

Chapter 1

Introduction

1.1 About Arduino

Arduino is a microcontroller-based open-source electronic prototyping board that can be programmed with an easy-to-use Arduino IDE. Arduino consists of both a physical programmable circuit board and a piece of software the Arduino IDE uses a simplified version of C++.
They can be powered from either a 9V battery or a power supply, via a USB connection to the computer. It can be commanded from a computer or programmed by a computer and then disconnected and allowed to operate independently.
Johnny-Five is compatible with different types of microcontrollers and microprocessors among the compatible Arduino boards. The compatible Arduino boards are Arduino UNO, Arduino Leonardo, Arduino Mega, Arduino Fio, Arduino Micro, Arduino Mini, Arduino Mini Pro, Arduino Nano, Intel Edison Arduino, etc.

1.1.1 Arduino Uno and its components

The Arduino UNO is one of the most famous boards in the Arduino family and is the best choice for beginners.

- USB Connector - It is a printer USB port used to load programs from the Arduino IDE to the Arduino board.The board can be powered through this port.

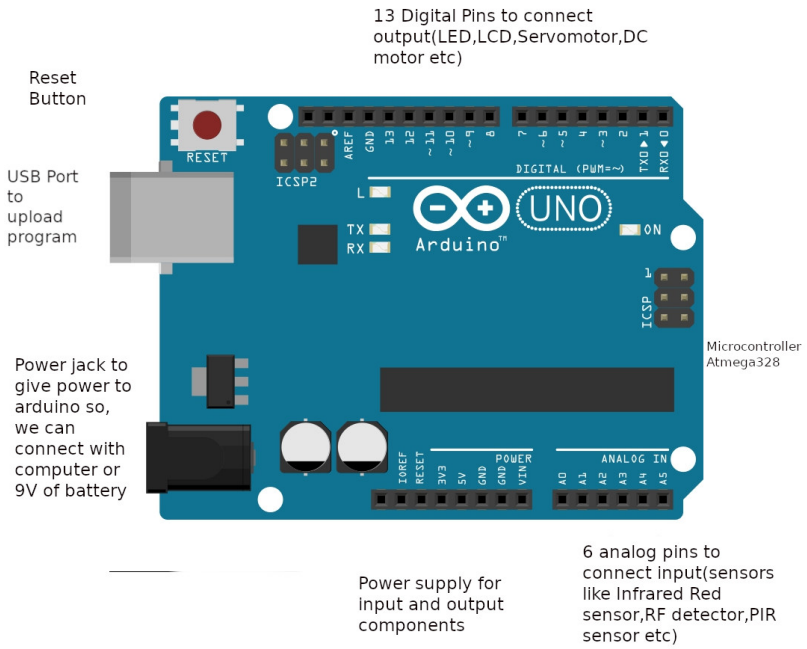


Figure 1.1: Arduino Uno

- Power Port - The Arduino board can be powered through an AC to DC adapter or a battery. The power source can be connected by plugging in a 2.1mm center-positive plug into the board's power jack.The Arduino UNO board works on a voltage of 5 volts, but it can withstand a maximum voltage of 20 volts.If high voltage is supplied to the board,then there is a voltage regulator that protects the board from burning out.

- Microcontroller -The microcontroller used on the UNO board is the ATmega328P by Atmel. Atmel is leading microcontroller manufacture. This is the most mainly visible black rectangular chip with 28 pins. It is supposed to be the brain of Arduino.
  Atmega328P has the following components in it.

  - 32 KB of Flash Memory : Programs loaded from the Arduino IDE are stored in flash memory
  - Ram of 2KB : This is runtime memory.
  - CPU : It controls everything that happens within the device. It receives program instructions from flash memory and runs it with the help of RAM.

  - Electrically Erasable Programmable Read only memory(EEPROM) of 1 KB : It is a type of non-volatile memory, and it holds data even after restarting and resetting the device.

  The Atmega328P is pre-programmed with a bootloader. IT allows us to upload a new Arduino program directly to the device without using an external hardware programmer, making the Arduino UNO board easy to use.

- Analog Input Pins : The Arduino UNO board has six analog input pins labeled analog A0 to A5. These pins can read signals from analog sensors such as temperature sensors and convert them into digital values for the system to understand.These pins only measure voltage and not current because they have a lot of internal resistance. So, there is only a less amount of current flows through these pins. However, these pins are labeled analog. These pins can also be used for digital input or output.

- Digital Pins: These pins with digital labels 0 to 13, can be used as input or output pins. When these are used as outputs, then these pins act as the power supply source for the components on which these can be worked. When used as input pins, they read signals from the components connected to them. When these pins are used as output pins they supply 40 milliamps of current at five volts which is more than enough to light an LED. Some digital pins are labeled with a tilde symbol next to the pin (PINs 3, 5, 6, 9, 10, and 11). These pins act as normal digital pins, but can also be used for pulse width modulation PWM that simulates analog outputs such as fading and LED in and out.

- Reset Switch : When this reset switch is clicked, then it sends a logical pulse to the reset pin of the microcontroller and now triggers the program again.

- TX RX LEDs : TX means to transmit and RX means to receive. These indicators are the indicator LEDs that blink whether the UNO board is transmitting or receiving data

1.2 Need for an Arduino Simulator

Students or users are using Arduino Board to purchase, or they have previously visited university or school labs and then discovered some other components that are required

to use these components but are not available or suitable for all students or users. Students don't have Arduino gadgets to use at home. These are the difficulties that they have to face in the early days. This is why we prefer to choose a simulator - the Arduino Simulator is a virtual representation of a real Arduino circuit. Here, using the Arduino Simulator, we can create a circuit with a virtual Arduino board. We can use the same code as we used the real Arduino board. We can learn to program as well as learn how to create circuits from anywhere where there is a computer system. We no longer need to connect to equipment in the workshop. We can do it anywhere. Second, it is much easier to track down hardware and wiring errors in the simulator. It can be difficult to visualize where the wires are connected and what pins are on the loaded Arduino and related components. But if we do this in the simulator, we will recreate it. goes much smoother in the real world.

1.3 Existing Arduino Simulator

There are already works that are not based on the web. Part of the work involved with the existence of the Arduino simulator work is a simple button as an input and an LED as an output. The other is a simple DHT11 component and an LCD that measures and displays temperature and relative humidity on the LCD.

1.4 Advantages of Web-Based Simulators

Web-Based Simulators are calls to computer simulation services on the World Wide Web, usually through a web browser. The Internet is increasingly seen as a medium for providing modeling and simulation applications, and therefore an area of research is emerging in the modeling community. In the case of a Web-based simulator, this can be an alternative to installing software on the student's computer. Web simulators can happen both server-side and client-side. In server-side simulations, numerical calculations and visualization (creating graphs and other computer graphics) are performed on a web server, while an interactive graphical user interface (GUI) is often partially provided on the client-side.

Students or users do not need to worry about how old their computer is or recommended hardware requirements such as hard disk or solid-state drive space requirements, CPU processors, and RAM. It is an operating system-independent web-based

simulator. In the Web-based Simulator, their virtual environments can be simulated on old home computers, allowing users to play or work in a web browser. We also have a built-in code editor for writing code in it.

Some web simulators run simulations with amazing features. Tinkercad Circuits is a free online service from Autodesk launched in 2017 and is without a doubt the most user-friendly web simulator. We can easily design our circuits, create a program in block or text format, and then eliminate it. Arduino cards and I / O interfaces, and interacting with the code works like a charm, and the code can be downloaded and shared with other developers.

However, there are some limitations, tinkercad has functions to add components and parts and create your circuit, but does not have functions to add other Arduino libraries or APIs to it. We hope that in the future we can see these missing features on it.

1.5 Need for an Application Programming Interface (API) for Arduino Simulators

For Simulation, the Arduino API is a tool for exchanging data between the Arduino and individual external systems. When using the API, the client application reads or transmits data to the Arduino board.
Let's say simple Arduino workflows like an LED connected to an Arduino board. We are creating a simple website to control the Arduino UNO board. There will be a node js server that will allow us to communicate directly with USB devices or serial ports from the web browser to the Arduino board. We have to enable a signal or flag to use it through a web browser. Using the Johnny-Five library will connect the implemented API to the Arduino and its components and allow using its API to turn LEDs on and off in the simulator. This is the requirement that I am going to show in this advanced functionality project.

1.6 Aim and objective of the projects

During the covid pandemic, students who do not have an Arduino board and components cannot buy an Arduino board because some areas are blocked. There are no stores to buy Arduino. So for the sake of learning, they can't go out and buy

an Arduino board. Some students who lived in the area have shops open after the lockdown, but they don't have a big budget to buy an Arduino board with their components. In this situation, this project aims to teach students how to learn it using the Johnny-Five library. This library will link the provided API library to the Arduino and its components. Students can access Arduino and other devices without touching them using the simulator. After using the johnny Five library, students will have a basic understanding of the Arduino board simulator and its functions, as it is free source code, and they will have the opportunity to implement their creative ideas and develop. libraries.
The objective of the project:

- Create schematics with Proteus Design Suite.
- Importing and using the modules such as Johnny-Five as well as socket io and express.
- Importing and using modules like Johnny-Five and socket io and express.
- Connecting components using Proteus Design Suite.
- Implementing the Johnny-Five API via Node JS.

1.7 Organization of the thesis

In this project, I have done some research on the Arduino board, its components, as well as the use of Johnny Five's API.

Chapter 1 : This chapter gives an overview of the Arduino board. One of the most popular Arduino boards is the Arduino Uno, and it is one of the most famous boards in the Arduino family. In this project, we will also look at its components such as USB connector, power supply, microcontroller, analog pins, digital pins, reset switch and TX RX LED. So, before starting this project, we must read Arduino and its components. We will find out why we need an Arduino simulator. If we use it, we can use it without buying an Arduino or attending workshops. We will also look at why it is necessary to use the Application Programming Interface (API) for the Arduino simulator and using the API, we can perform a specific task for the Arduino in the simulator.

Chapter 2 : In this chapter, we will also explore the design requirements are, we will be able to see what programming skills are required to use in this simulation and what are the schematic design requirements when modeling, as well as simple information about the components that will be used in the simulation to make a custom an interface to make it a web page to control the Arduino in simulation. We can also read what are the requirements for the tools and technologies used in the simulation..

Chapter 3 : In this chapter 3,by examining how the Arduino board works, we can also learn how its components such as resistors, input and output components, and their correct wire connection to the Arduino board, power supply, and ground. In the next section, we will also be able to explore the API, how the API will connect to the Arduino board to complete the task. Let's figure out what Johnny-Five is and why the Johnny-Five API is so good for working with Arduino. And how Johnny-Five works very well with socket io and express library.

Chapter 4 : In this Chapter 4,we can also look into Proteus Design Suite, and what is it? We will also be able to learn how to use Arduino, input and output components, and two other Arduino boards via wire to Arduino in this simulation software.

Chapter 5 : Then we can also see the tools and technologies used in the simulation, such as com0com to create a virtual serial port. We also read to take advantage of the importance of ArduinoStandardFirmata, Node js, and Visual Studio code, and how we implement johnny-five's code with VS Code tools used in modeling.

Chpater 6 : In this chapter,finally, we can see the simulation run and see how I run and finally what the result is after that.

Chapter 2

User Requirements

2.1 Introduction

From the point of view of future beginner users, the simulator supports simple communication to control, even write a program for the Arduino Board. Previously, students had to think about buying an Arduino Board or they need to go to a workshop for practical use. Now there is a Simulator, it can be used for all students, and they do not need to buy an Arduino and additional components if they want to use it for educational purposes. Any user can learn from it, be it a high school student, college student, or faculty member.

Users must know programming in C, C ++ and Javascript - these are the languages in which we must write the code for the Arduino board according to this project.
Users only need a Windows system to use the simulator. The simulator to be used is Proteus Design Suite, so we have to remember.
By reaching out to community support, a large amount of community will be helpful for users. Since we are using Johnny-Five's Javascript-based framework, it will be easier for novice users. Users must obtain information from Johnny-Five community support if they get stuck with any issue.

Now that we get to use the wire connection between the Arduino and its components. Therefore users don't have to worry about how to do it. Users can get help from this project. Thus, they can get any information on components from this project.

Users has also knowledge on HTML,CSS,since we are using web browser where one webpage will communicate to the Arduino board running in the simulator through any virtual serial port.Users have to build a webpage where there will be control of this Arduino.With Respect to that ,Users must to know HTML,CSS,in order to built it.

Users also know HTML and CSS as we are using a web browser in which one web page will communicate with the Arduino board running in the simulator through any virtual serial port. Users need to create a web page that will control this Arduino. In this regard, users need to know HTML, CSS, to create them. Now that we get to use the wire connection between the Arduino and its components. Therefore users don't have to worry about how to do it. Users can get help from this project. Thus, they can get any information on components from this project.

2.2 Functional Requirements

- **Pull up and Pull down resistors** - In digital circuits, we often want to supply input using switches or push-button when doing that we are running into the risk of leaving the input pins into a floating State is where in footprints of a chip cannot detect input logic and ultimately lead to unexpected output, To avoid this floating State resistor of a specific value is used these are known as pull-up or pulldown resistor. Pull-up resistor connect the input pin to VCC making it to read logic one , pulldown resistor connect an input to the ground making it to read logic zero.

- Calculate the discharge time of a battery.

- **Watch out for resistance Wattages** - The wattage rating of a resistor means the maximum amount of power a resistor can safely dissipate in the form of heat. If the dissipation in the resistor exceeds the maximum wattage rating, then the resistor is likely to fail and burn out.

- **Using transistor pairs/arrays** - Transistors are widely used to turn on or off the load and amplify signals. When applying a single transistor in the circuit, where the transistor could not provide enough current to turn on the load. In such cases, we could use the Darlington transistor.

- **Using PWM signals to save power** - PWM is a modulation where the duty cycle of the pulse is modified using this PWM signal.

2.3 Non-Functional Requirements

- When simulating an Arduino board, the web page that controls the Arduino board should be loaded within 3 seconds.
- The process of completing any task controlled through a web browser should be done quickly on the Arduino board simulator.
- The Arduino board should respond quickly or update its status to the user via a web browser.
- Make sure the system is compatible with the simulator. In this project I am using the Proteus Design Suite simulator.

2.4 Designing Considerations

- **Accessibility** - We must design Arduino and its components in a way that is accessible and learnable by users or students.
- **Efficiency** - We need to design a way to efficiently use CPU power and resources. Increasing the number of Arduino and its components can lead to maximum use of CPU power and resources.
- **Extensibility** - We must develop such designs that in the future can be expanded by other projects.
- **Performance** - We must design so that it does not have any performance impact, such as speed or accuracy.
- **Durability** - We need to ensure that the design is durable during the simulation and not disrupted during the simulation.
- **Usability** - Design it to make it easy to simulate.

2.5 Software Technologies

- **Needs of Arduino IDE** - Depending on the design requirement, we have a two-slave Arduino board. I've written program code for both the slave Arduino board, so the software requires the **Arduino IDE**. To write Arduino programming with this software is divided into three sections.
  - **define** : Define pins with variables.
  - **void setup** : Define pin as Output or Input using Pinmode().
  - **loop** : Here we write the main program, command, and logic.

  So, we will definitely see schematic designs for Slave Board 1 and Slave Board 2 in Chapter 4, as well as the programming code for Slave Board 1 and Slave Board 2 in Chapter 5.

- **Tools for creating a serial port** : Based on the design requirements, we can see that the compim component is being used. Compim simulates physical serial communication and transfers it to the circuit as a digital signal. The serial ports of the computer will be used to transfer all serial data outgoing from the CPU or UART model. A virtual serial port can also be created using software called **com0com**. The software allows for multiple virtual COM ports and provides an excellent opportunity to fully simulate the behavior of a serial port. It will connect the Arduino board to the system via the serial port. Since we are using the compim component in the schematic design and also going to use Johnny-Five, we need to use the com0com software, so we will see this software in Chapter 5.

Chapter 3

Working of Arduino Board

3.1 Introduction

Arduino Uno is a microcontroller board based on the 8-bit ATmega328P microcontroller. Along with the ATmega328P, it run with other components, which are input and output components. The Input and output components allow the Arduino to communicate with the outside world by moving data in and out of the system. The input component is used to transfer data to the system. The output component is used to send data from the system. And we will learn about important aspects of APIs on Arduino and how Johnny Five's library is really useful for writing code for Arduino.

3.2 The Components of the Arduino

3.2.1 Resistors

Resistors resist the flow of current in a circuit.The bigger the number of the resistor,the more current they resist.

Figure 3.1: Two Resistors with 220 ohm and other 250 ohm

When it resists current,the resistor takes that electrical energy and transforms it into heat energy.Therefore,it's important to know that resistors can get hot.

3.3 The Output Components

3.3.1 LEDs

The LED's legs are connected to two pins on the Arduino that is ground and pin 13. The component between the LED and the ground is a resistor, which helps limit the current to prevent the LED from burning itself out. Without it, It does not matter whether the resistor comes before or after the LED in the circuit, or which way around it goes. The colored stripes identify the resistor's value, and for this circuit, anywhere from 100 ohms to 1000 ohms will work great.

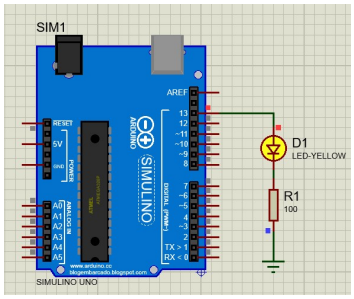


Figure 3.2: Arduino is connected with a led and resistor with 100 ohms

The LED , on the otherhand , is polarized,which means it only works when the legs are connected a certain ways.The positive leg , called the anode , usually has a longer leg, and gets wired to power , in this case coming from our Arduino's output pin.The negative leg called the cathode , with its shorter leg, connects to ground.

3.3.2 Fan motor

The fan motor feet are connected to two pins on pin 3 of the Arduino and ground. The component between the DC motor and the ground is a 100-ohm resistor. If any voltage is supplied to it, it rotates clockwise or counterclockwise.

3.3.3 Servo Motor

The servo motor pins connected to the three pins on the Arduino are ground, pin, and VCC (common collector voltage is the power supplied to the components). It is a type of PWM motor that we can turn or stop at a specific angle. Here it is connected via a wire to pin 10 of the Arduino board.

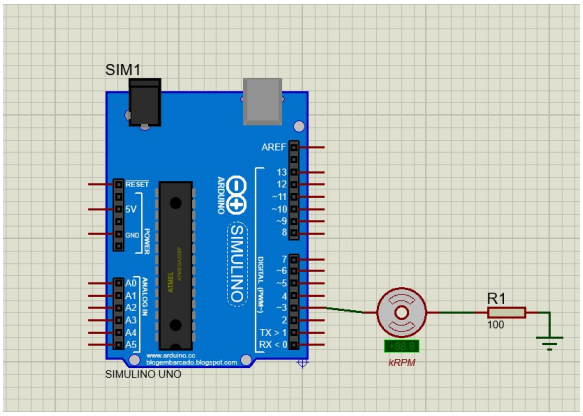


Figure 3.3: Arduino is connected with a fan motor and resistor with 100 ohms

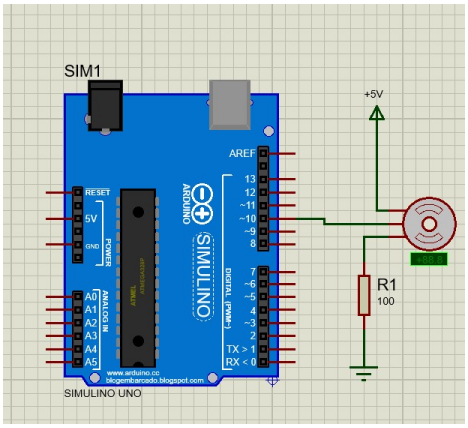


Figure 3.4: Arduino is connected with a servo motor and resistor with 220 ohms

3.3.4 LCD

It is a liquid crystal display, this can represent characters on it. For example sensor's value or someone's name etc.

3.4 The Input Components

3.4.1 TH02

It measures temperature in Celsius, Kelvin, and Fahrenheit. It also detects relative humidity.In the figure 3.6, the legs of TH02 connected to three pins on the Arduino are ground and two are analog pins. TH02 only works with Arduino analog pins A4, A5. On the Arduino board, analog pin A5 is used for the data line (SDA), and analog

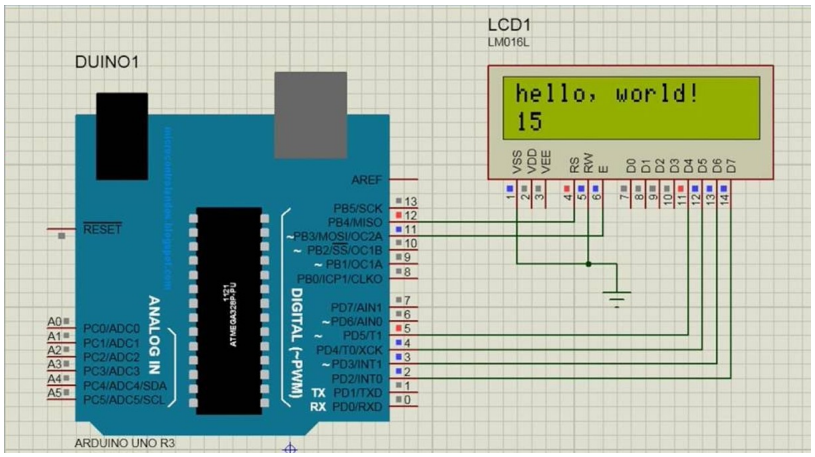


Figure 3.5: LCD connected with arduino

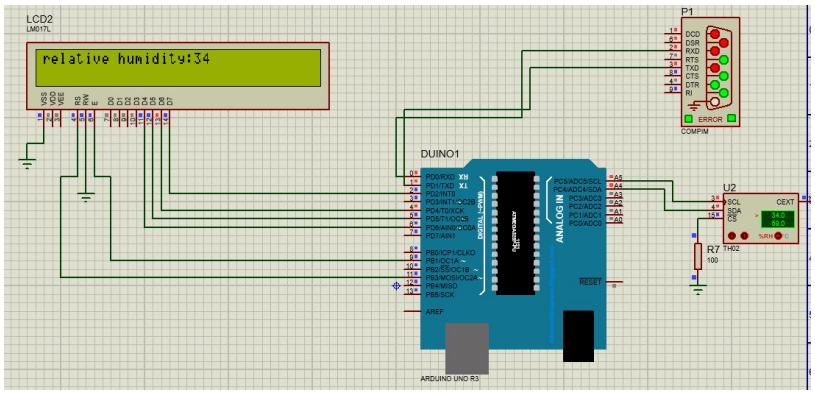


Figure 3.6: One TH02 components with LCD displaying relative humidity

pin A4 is used for the clock line (SCL). It can display temperature as well as relative humidity on LCD.

3.4.2 Push Button

The push-button legs connect to two pins on the Arduino, ground, and the pin or analog. When we press the button, it will close the circuit. When we release the button, it will open this circuit.Pressing the button turns the fan motor on and off. In figure 3.7,it is connected to pin 13 of the Arduino. The fan motor is connected to pin 3 of the Arduino board.

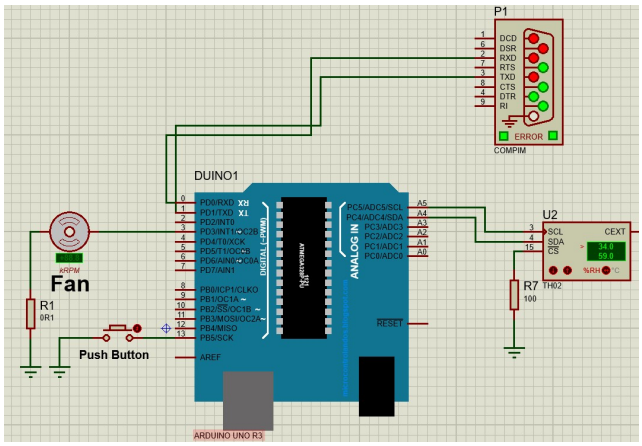


Figure 3.7: Pushbutton to turn On or Off the Fan

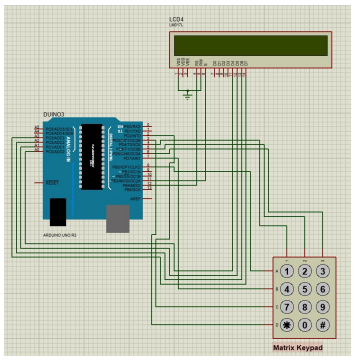


Figure 3.8: Matrix keypad connected to Arduino

3.4.3 Matrix Keypad

It is the layout of the buttons in the XY matrix. It is a standard pushbutton switch. X-Y conductors are connected by pressing a button. Only one press key is accepted at a time. Here, pins A, B, C, D of the X Matrix Keypad wires are connected to pins 8,7,2,3 of the Arduino and pins 1,2,3 of the Y Matrix Keypad wires connected to Pin of 4,5,6 of the Arduino.Here, with this connection, whichever button pressed will be displayed on the LCD.

3.5 Arduino connecting to an API

The software or its elements do not require a graphical user interface to communicate with each other. Software products exchange data and functionalities through machine-readable interfaces. – APIs (application programming interfaces). (Fig. 5.8).

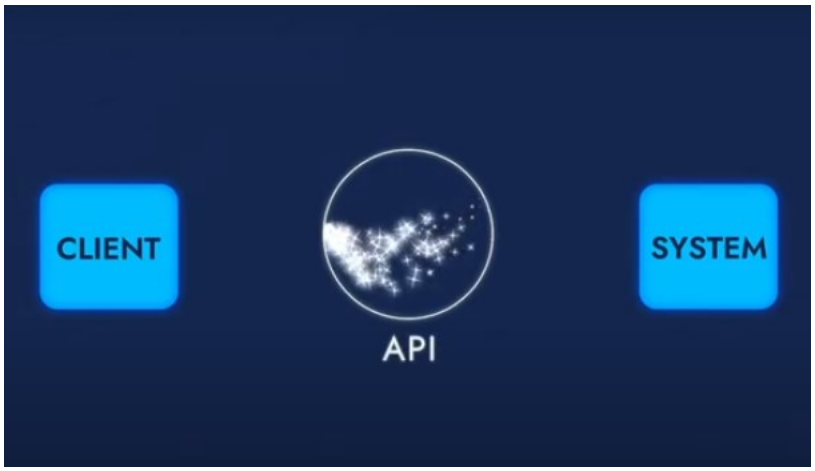


Figure 3.9: API between Client and System

When we use a web browser on your system, the application connects to the Internet and sends data to the server. The server then retrieves that data, interprets it, performs the necessary actions, and sends it back to the client-side. The application then interprets that data and presents us the information we wanted in a readable manner

When we hang around programmers, we might have heard about APIs and how they can be used to perform certain tasks or retrieve some data. But what are these exactly, and why were they created? We can understand the problem with a simple, non-IT-related example. When we go to a restaurant and order some food, we interact with the waiter. We can order food and drinks, ask questions about the menu, request and pay the bill, and much more. In this example, the waiter is shielding us from all the complicated stuff that happens behind the scenes. We don't have to worry about stoves, ovens, dishes, managing stock, or pouring drinks. He is the interface between you and all of the services that a restaurant offers. Giving you a way to interact with the restaurant while still shielding you from all the complexity behind the scenes. In a way, the waiter can be seen as the API of the restaurant, and through this example, we intuitively understand why they are useful. The term API stands for Application Programmable Interface, and it's a way for different programs to work together in various ways. There are many types of API's and reasons why they are used.

Similarly, in this project, we are going to see that the API is saving us from all the complicated stuff like Arduino workflow and other tools associated with it. In terms of users who are supposed to control the Arduino, they don't need to worry

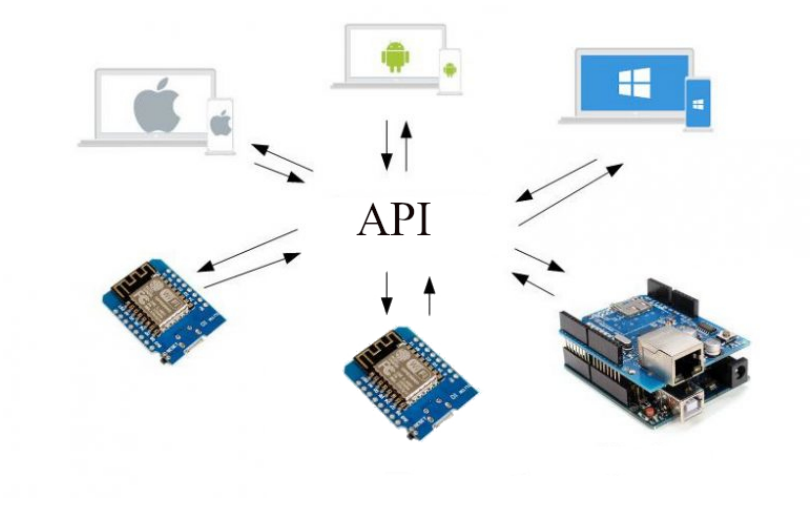


Figure 3.10: API between Arduino and System

about how the Arduino and its equipment will do their job. Users are allowed to control the landscape of the Arduino. Here, the API is the intermediary between the Arduino and the client, which is used to exchange data between the Arduino and other external systems.

In the next section, we are going to look at the johnny-five framework, where we will learn how its API will work with arduino.

3.6 Installation

To install Johnny-Five library for Arduino Simulator, install the module with npm install johnny-five via Windows Terminal or Visual Studio Console window.

```
C:\Users\Karan Soren\Documents\UoH\Sem_6\Project\Report_Sample\sample>npm install johnny-five

added 91 packages, and audited 92 packages in 2m

5 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities

C:\Users\Karan Soren\Documents\UoH\Sem_6\Project\Report_Sample\sample>
```

Figure 3.11: installing johnny-five

The Johnny-Five libraries work well with popular application libraries such as express.js and socket.io.

```
C:\Users\Karan Soren\Documents\UoH\Sem_6\Project\Report_Sample\sample>npm install socket.io

added 20 packages, and audited 112 packages in 12s

5 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
```

Figure 3.12: installing socket io

To install socket-io, install the module with npm install socket.io via Windows Terminal or Visual Studio Console window. Socket io is a library that provides a mechanism for communication between the client browser and the server. And this communication is bi-directional and that means that data can be transferred from client to server as well as from server to client. socket io is always open, which allows real-time data transfer in our application. So this is all amazing, but what does this mean in a real example

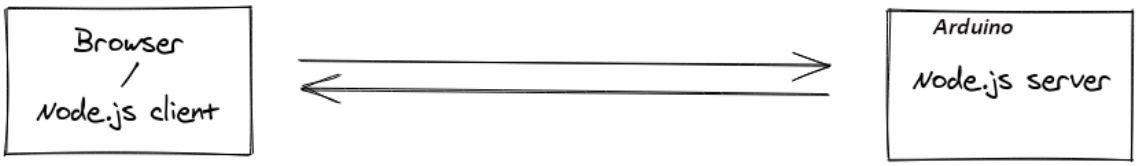


Figure 3.13: client sending and requesting data from Server

Imagine we have some kind of web application that is hosted on the server via Node Js. So this system presents Arduino workflow scripts to bypass this system. Users submit any Johnny-Five API request from the client side through their web browser. Suppose I turn on an LED through the Johnny-Five API and this request data is sent to the Arduino server immediately. Consequently, data can be transferred between them in real time via a web browser.

Express JS is a very powerful tool for writing server-side code for any web application in this project.It allows you to configure middleware to respond to HTTP requests and also allows you to dynamically render HTML pages. To install express, install the module using npm install express via Windows Terminal or Visual Studio Console window.

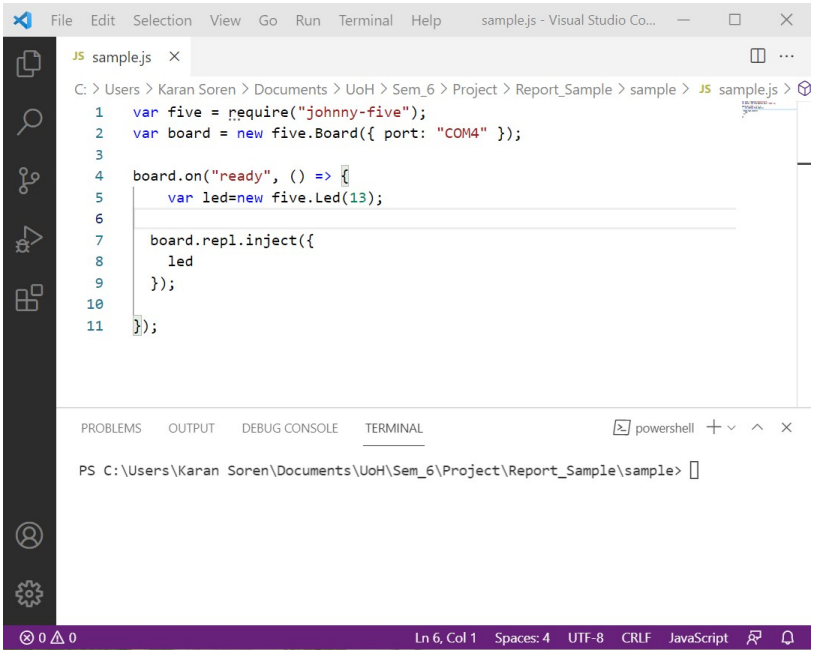
Before getting started with Express, make sure we know HTML with javascript So that we can write the javascript code that we will be working on in this project

```
C:\Users\Karan Soren\Documents\UoH\Sem_6\Project\Report_Sample\sample>npm install express

added 50 packages, and audited 162 packages in 12s

5 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
```

Figure 3.14: Installing express

in Express. It is javascript technology which means javascript language that we will use in this project to write code, besides that, we also need to know node js like all express codes. We will be working on this project only in the node environment. So, make sure we know npm and node as well.

3.6.1 Johnny-five



Javascript is designed to be asynchronous. When we launch the browser, we do something based on the user's asynchronous actions and even directed clicks. This translates very well into robotics, because usually what happens is we have many different sensors and asynchronously. We receive an input signal from the sensor, we act with the robot, if we look at the standard program. We know a robotics-based

program, and we end up writing our event. So we do our initial setup and then loop and always keep polling for specific values. JavaScript provides a nice abstraction of this concept, where we can simply say no matter what the event occurs. Now take our actions like a robot.So how does it work? Johnny-Five is a Javascript library for controlling the Arduino, but it cannot run on the Arduino, so we need a separate host computer and firmata which loaded into arduino, are basically serial control mechanism for microcontrollers, so we load this on Arduino, and then we connect the host computer to run it, and here we will see that I am using an Arduino Uno, so have we have a very compact package, what else is interesting.So, we require the johnny-five framework that is fast enough to perfectly communicate with the Arduino in real time, so I accepted the johnny five library for this project.
Johnny-five library is a javascript robotics and IoT platform which allows us to communicate with micro-controllers using node js.
It has been actively maintained in a GitHub repository created by a gentleman named Rick Waldron. Initially, Johnny Five could only work with the Arduino microcontroller through the Firmata protocol. Now, it allows us to work a Tessel or Raspberry Pi or any different device.

Johnny Five gives us helper methods that make it easy to do amazing things with our hardware. We can write code using Johnny-Five and run it on different devices.Starting with the Johnny-Five, we'll take an LED and connect it to the Arduino and turn its LED on and off.

```
>> led.stop() // to stop blinking
then
>> led.off()  // to shut it off (stop doesn't mean "off")
then
>> led.on()   // to turn on, but not blink
```

Figure 3.15: Led on and off using Johnny-Five

In line number 1, it is loading and caching the Johnny-Five module. Then, the Arduino board is initialized and connecting to the system via COM4. Now the board is ready with pin 13. Here, pin 13 is defined for the LED. The scripts for turning the LED on and off are led.on() and led.off(). This script will make available() and off()

functions available in the REPL.This makes it possible to type these scripts in the terminal and the execution of the LED will be done.

3.7 Summary of the Chapter

In this chapter, we read about Arduino components such as a common component such as a resistor, and input and output components. We will learn the importance of the resistor used with each component to connect through a wire to the Arduino board. We read how to connect with a wire from output components and input components to the Arduino board in Proteus Design Suite. We also read about the API capabilities used in the Arduino board. And we are exploring the Johnny Five library and using its API. And why we chose this when modeling. We also read that Johnny-Five libraries work well with popular application libraries such as express.js and socket.io.

Chapter 4

Proteus Design Suite

Proteus design suite is a simulation software for Arduino which contains various tools for the schematic design and schematic drawing.It is mainly used for manufacturing PCB (Printed Circuit Board) designs.

4.1 Schematic Design using Proteus

I have simulated with the help of using the Johnny-Five library. The following programs that I used during the construction of the Arduino simulation.The list of tools I have used are as follows:-

- Three Arduino UNO
- Resistors of 100 ohms
- Compim
- Three Fan-DC
- Yellow led
- Green led
- Red led
- Blue led
- Three Servo motor

- Threee LCD (LM016L)
- TH02
- Keypad

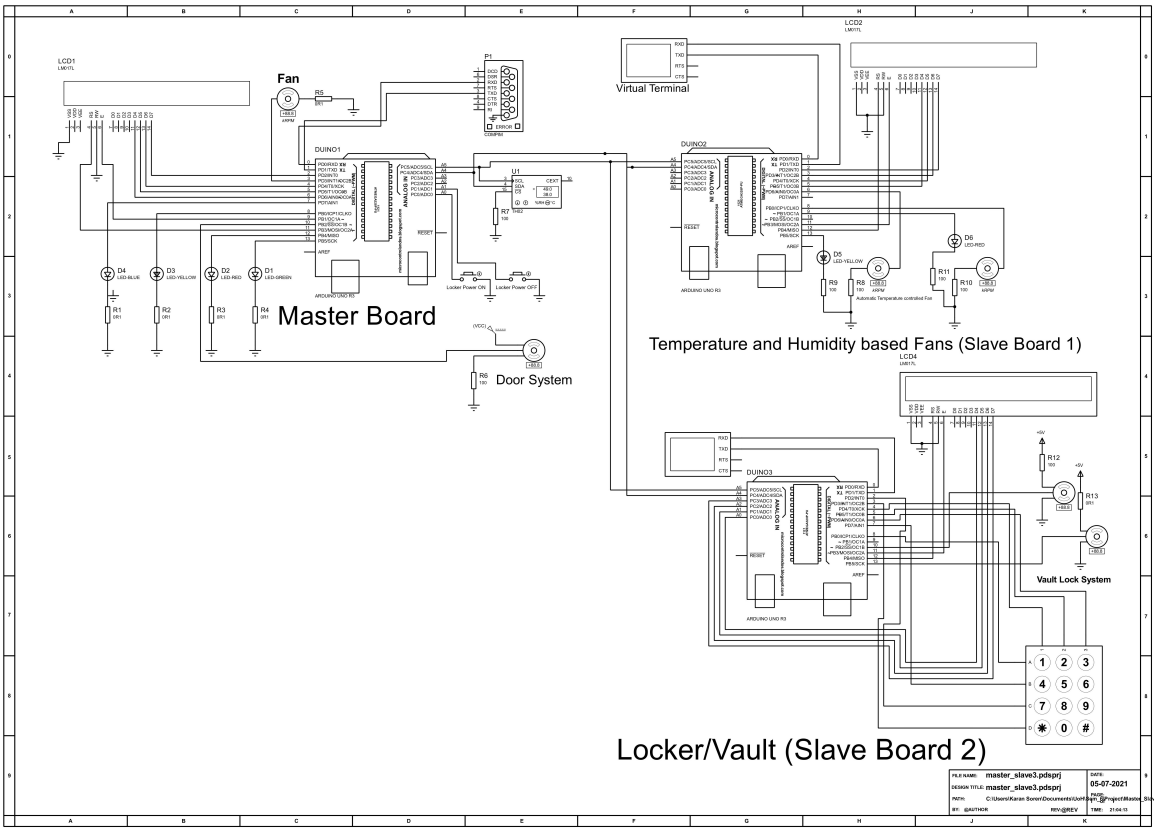


Figure 4.1: Schematic design

I have designed a schematic that simulates the LED, fan and door during the simulation. The TH02 sensor also measures relative humidity and temperature. Relative humidity and temperature rise and fall are displayed on the client-side while changes through this sensor. The LCD (LM017L) displays the working of the Arduino board. i.e. if I turn on the toggle switch of the yellow-LED, the LCD screen will display that the yellow LED is on.

4.1.1 Master board and Slave board 1

I used three Arduino boards, one as the master and the other two as slaves. A connection is established between the boards using the I2C communication protocol. I

used three Arduino boards, one as the master and the other two as slaves. Communication exists between boards using the I2C communication protocol. The master board will instruct the slave boards, the slaves will do their job. The slave board will display what has been done by the master or slave board. For example, if I turn on the yellow LED, the LCD displays "Yellow LED on" where it is connected to the slave LCD board. Two DC fans and two LEDs are connected to the slave board. If the temperature is over 38 ° C when TH02 (Digital RH Temperature Sensor) is connected to the mainboard, then turn on the fan and yellow LED. If the temperature drops below 39 ° C, the DC fan and yellow LED will turn off. The mainboard sends temperature data to the slave, the slave does its work according to the received temperature. Likewise, if the RH is over 50 percent when TH02 (Digital RH Temperature Sensor) is connected to the mainboard, the fan will turn on the red LED. If the relative humidity drops below 50 percent, the DC fan and red LED will turn off. The mainboard sends temperature data to the slave, the slave does its work according to the received temperature.

The Wire library enables communication with I2C devices, which are usually also referred to as "2-wire" or "TWI" (two-wire interface) devices.
This library enables an Arduino board to regulate Liquid Crystal Displays (LCDs).

4.1.2 Master board and Slave board 2

The master board is connected to another slave board. The second slave board is connected to a keypad, and two servo motors, and an LCD. The master board is connected by two push buttons that have on and off functionality to access the keypad. After pressing the push-button from the master side, the second slave board will have a chance to use the keypad, the user can enter the correct 4 digit password to unlock the locker/vault and the servo motors will run. These servo motors will be connected to the original door. So that the mechanism with their connection with the door helps to open it. One servo motor is attached to the door latch and the other servo motor is attached to the door. After typing the correct 4 digit password, the servo motor attached to the door latch will be opened first and then, the second servo motor attached to the corner of the door will be opened. If we want to lock the locker door, we need to press the button from master side. The servo motor attached to the

corner of the door will close first. And the second servo motor, which is attached to the door latch, will be turned off.

Chapter 5

Implementation and Code

5.1 API Methods

To simulate the arduino we are using the following tools and technologies.

- com0com
- Arduino Standard Firmata
- Node js
- Visual Studio Code

5.1.1 com0com

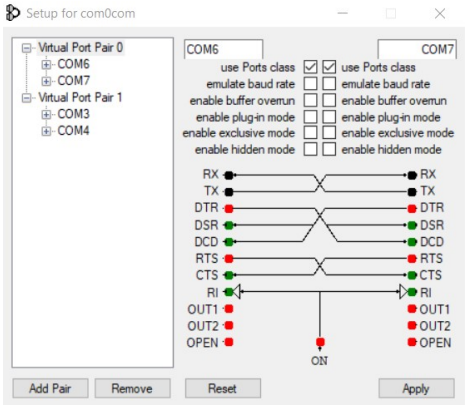


Figure 5.1: Creating a virtual port for the Arduino simulator and computer system

com0com is an application which virtual create serial port.We have created virtual serial ports for the Arduino board as COM3 and johnny-five js file as COM4.These

virtual ports will be used to connect the serial port component from the Proteus design Suite to the node.js that johnny-five uses. Using com0com, we can create and configure the required number of virtual ports. This way we can link more than one Arduino board or other components in the simulation. To properly configure virtual ports, and some useful commands to do so.In the figure 5.1 we can see the com0com setting, enter the name of the virtual port as COM3 and COM4. Click the checkbox on either side of the virtual port next to "use Ports Class". Then finally click Apply.Johnny-five uses the Firmata protocol to communicate with the Arduino, so we need to set the baud rate of our serial ports to 57600. To do this, open a command prompt and run the following line:

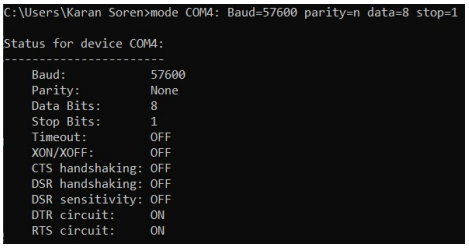


Figure 5.2: command for setting virtual port COM4

In this command, we also set some other configurations for our virtual serial port, such as parity and stop bit, just to be sure. Once the definitions for our first serial port are complete, do the same for "COM3".

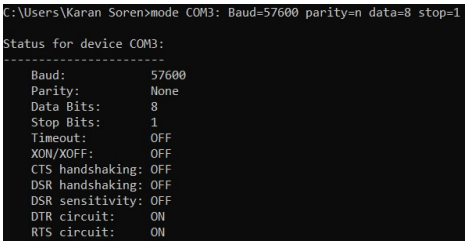


Figure 5.3: command for setting virtual port COM3

5.1.2 Arduino Standard Firmata

The Arduino Standard firmata protocol is a protocol that serves us so that we can communicate via Arduino with any other software or a connected computer.The Arduino standard firmata is built in such a way so that it enables us to make our Arduino

board can connect with different programming languages as such and thus make it much more versatile.

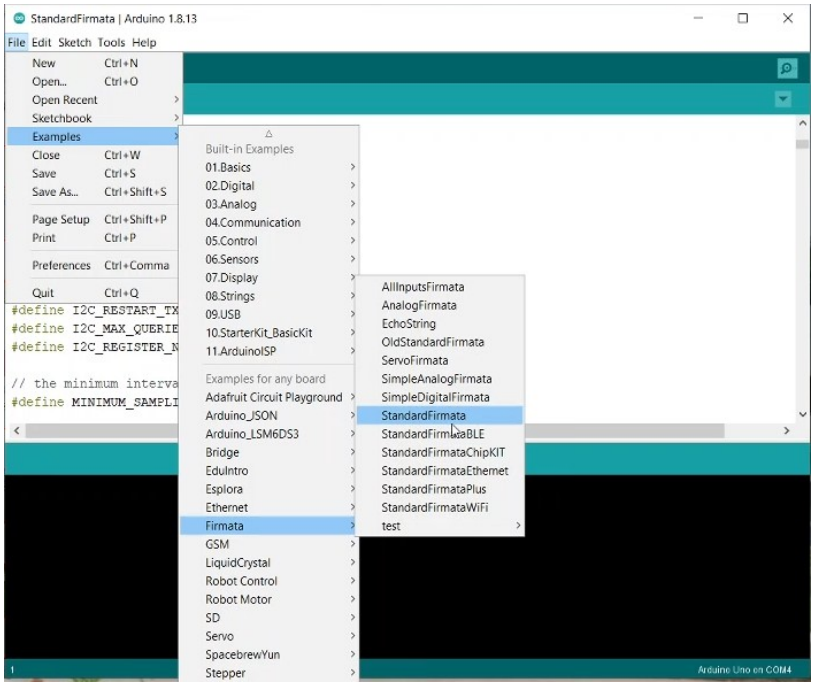


Figure 5.4: Getting Standard Firmata from Arduino IDE

The Arduino integrated development environment is a cross-platform application that is written in the programming language C and C++. It is used to write and upload programs to the Arduino board ,it can run on Windows Mac OSX and Linux. We can find different versions of the Arduino IDE.To get the firmata of basics downloaded in our Arduino Uno board we have to go to the file from Arduino IDE then to examples to look for firmata and then standard firmata then it will open up a file. Then already ready-made file will have to upload to the Arduino board just, click on verify we'll see the compiling sketch is in progress we can see the progress bar in the arduino console window. Once this is done we are good to upload this so we can click on upload. We can also see the uploading progress bar. Once that is done it will get notified there so this should be notified then uploading that means that our board is now ready to get communicated via johnny-five.

But in the simulation, we don't need to click on upload. After compilation, get those files that are created as standardFirmata with extension elf or hex files. Path of these files with extension elf or hex will be able to be seen through the Arduino ide console

window. Then, insert this hex or elf file into Arduino through Proteus Simulation. The Johnny-five library will communicate with the virtual Arduino board using the firmata protocol. In the next chapter, we are going to look at Proteus Simulation, where we will learn how to simulate Arduino using this simulation software.
To download the Arduino IDE, open a web browser. In a web browser, go to https://www.arduino.cc/em/software. Click Windows next to Download Options.

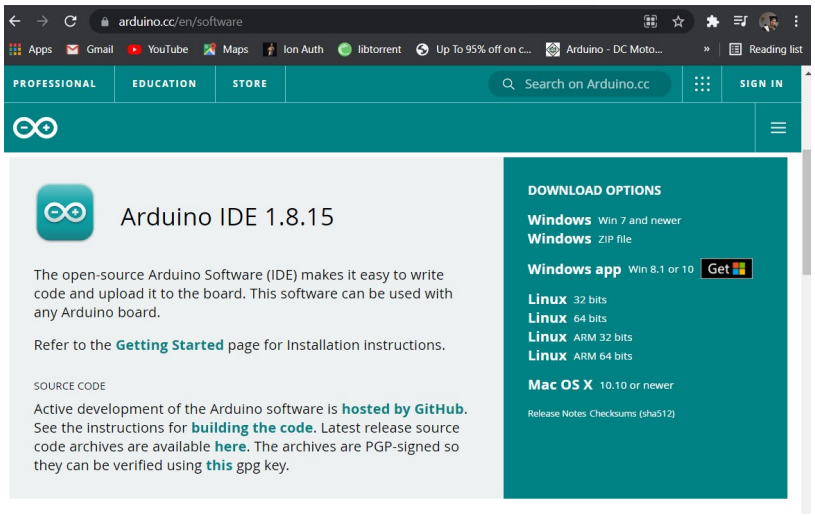


Figure 5.5: Downloading Arduino IDE

5.1.3 Node js

Node js is the runtime environment to JS outside the browser. It is developed by Ryan Dahl in 2009, and it's built on Chrome's V8 JS Engine.
Node js is not programming itself, but a runtime that allows us to run javascript on the server. We can see when Javascript first appeared in the 1990s. become more and more powerful.

In 2009 we saw the first release of node js, now until that time, it was impossible to write javascript code on the server. Most of the servers were written in languages like Java or PHP. This revolutionized web development because now a web developer could write a full-stack application in one language. So it takes to write Javascript on the server, whereas before it could only be written in a web browser.

With installation of node js,we will be able implement johnny-five modules in javascript js file.It has a large community support.

In a web browser, go to https://nodejs.org/en/download/. Click the Windows Installer button to download the latest default version.

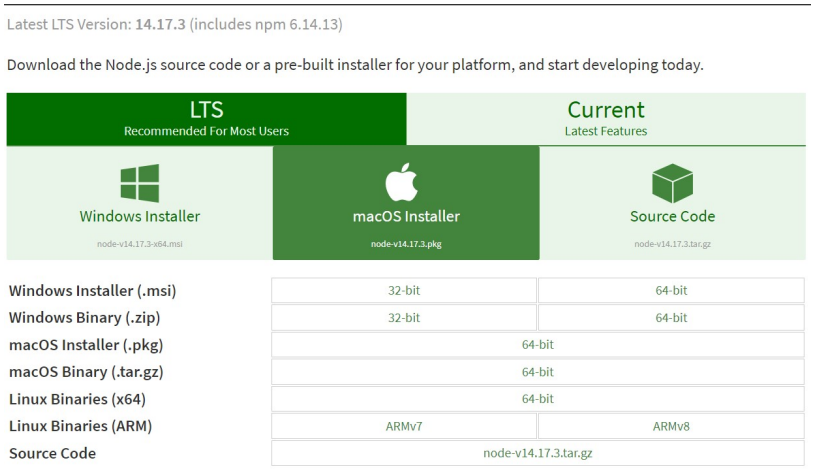


Figure 5.6: Downloading Nodes Js for Windows

The Node.js installer includes an NPM package manager.

- Once the installer finishes downloading, launch it. Open the download link in your browser and click the file. Or navigate to the location where you saved the file and double-click it to run.
- The system will ask if we require to run the program - click "Run".
- Welcome to the Node.js setup wizard - click Next.
- On the next page, review the license agreement. If you agree to the terms and install the software, click Next.
- The installer will prompt you for a location to install. Move the default location unless we have a specific requirement to install it elsewhere, then click Next.
- A wizard will allow you to select the components to add or remove from the installation. Again, unless we have a special requirement, accept the defaults by clicking Next.
- Finally, click the Install button to launch the installer. When it's over, click Finish.

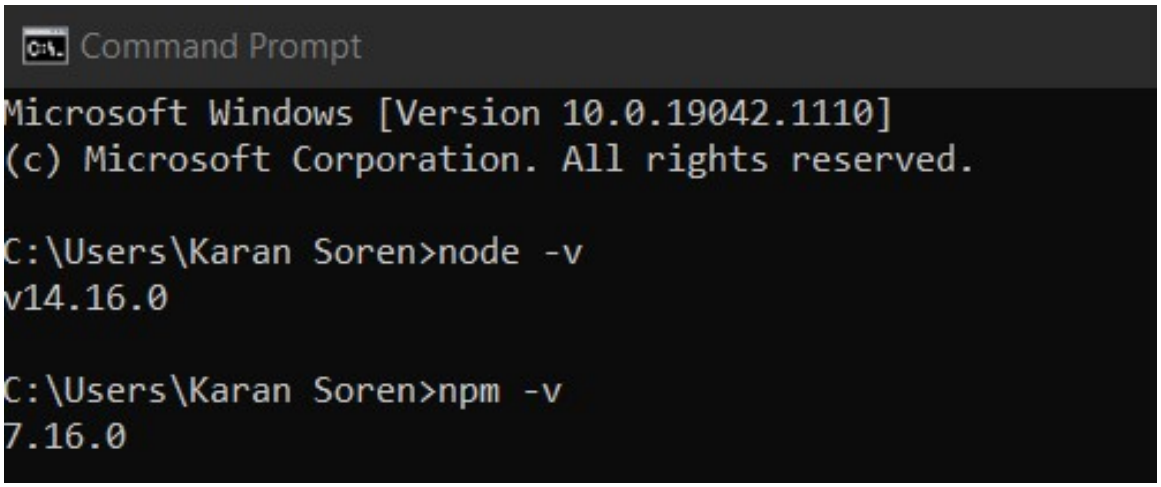


Figure 5.7: Command for verifying node and npm version

To verify installation, open command prompt or power shell and enter node -v

The Node.js installer includes an NPM package manager.In figure 5.7,the Command prompt display the version of Node.js installed on the system. We can do the same for NPM.

5.1.4 Visual Studio Code

Visual Studio Code is a code editor for all programming languages. It has inbuilt console functions. It is open-source application.

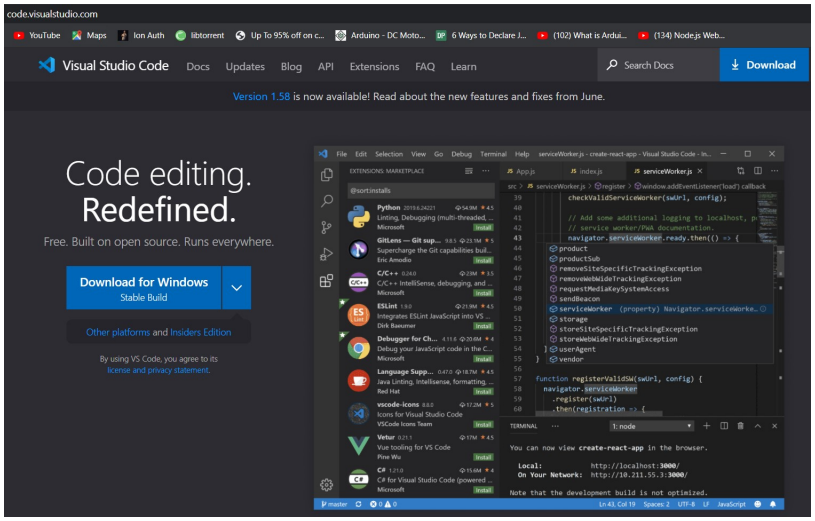


Figure 5.8: Downloading Visual Studio Code for Windows

In a web browser, go to https://code.visualstudio.com/ click "Download for Windows" to latest and stable version. With this Visual Studio code, we will write Johnny-Five code for the Arduino simulator and HTML code for the web page.

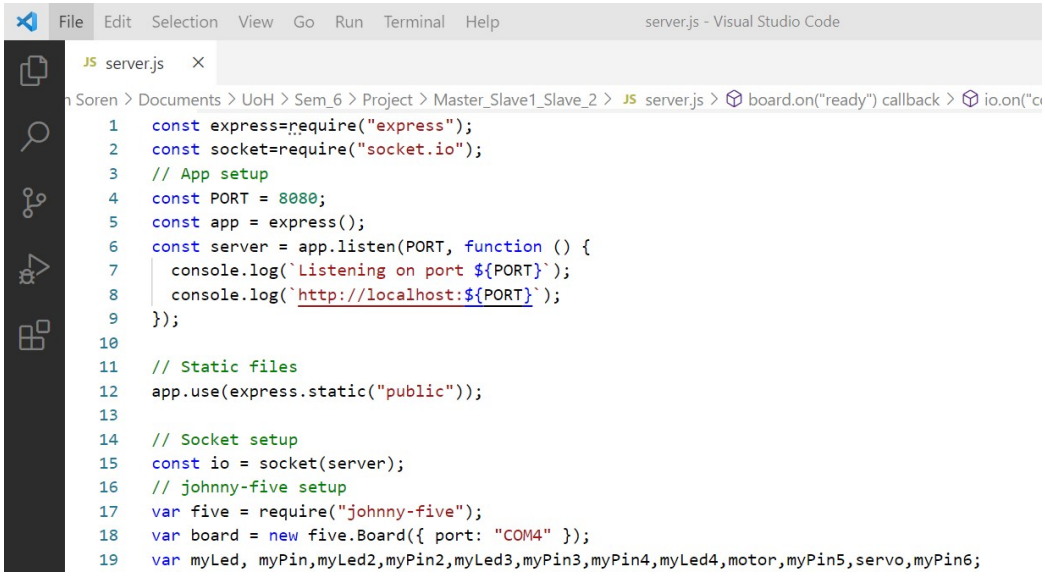


Figure 5.9: Johnny-five Code implementing from line 1 to line line 19

In figure 5.9 , On line number 1, the express module is imported. Express is a module that assigns functions, objects, or variables. Line number 2 imports the socket io module. Socket io is a library that provides a communication mechanism between the client browser and the server. Line 4 creates a constant variable PORT initializing with port values 8080. Line 5 creates a new express instance in the application with a constant variable. Line 6 app.listen () function is applied to connect and listen for connections on the particularized host and port with constant variable server. We can see that this application is starting the server and listening on port 8080 for connections. On line 12, it uses static middleware to serve everything that falls into this directory.It serves up HTML file in a directory named public. line 17 imports Johnny-Five modules and with this module we can use its API for Arduino. On line 18, we can additionally specify the port by defining it as a parameter property of the parameters object. Here, we specify it using COM4 for Virtual Serial Port. Line 19 declares variables for four LED, one motor, and one server motor.

lines 21–46, the board function is ready, and the variable declared on line 19 is instantiated for the LED and pin of the Arduino board.

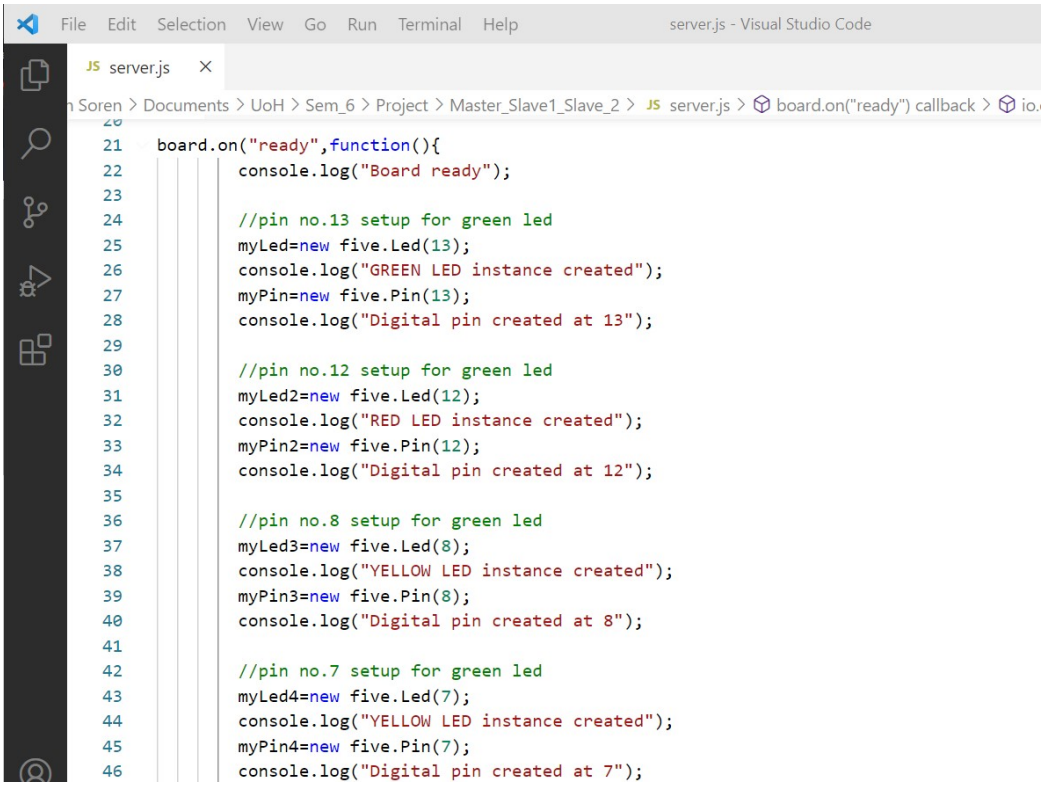


Figure 5.10: Johnny-five Code implementing from line 21 to line 47.

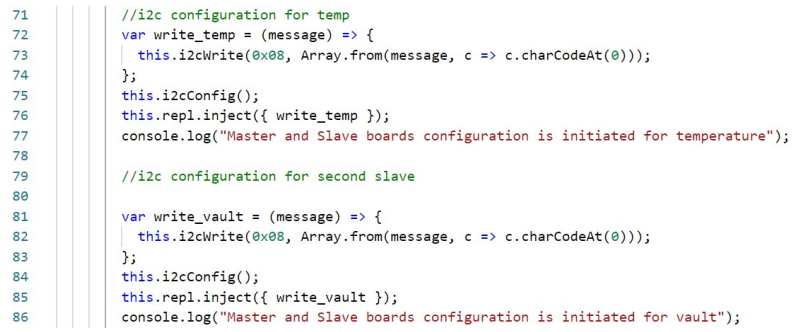
lines 48 through 58, the variable declared on line 19, are instantiated for the fan

```
48 //pin no.3 setup for fan motor
49 console.log("Digital Pin created at 3.");
50 myPin5=new five.Pin(3);
51 motor =new five.Motor(3);
52 console.log("Motor instance created");
53
54 //pin no.10 setup for servo motor
55 console.log("Digital Pin created at pin no. 10.");
56 myPin6=new five.Pin(10);
57 console.log("Servo instance created at pin no. 10.");
58 servo = new five.Servo.Continuous(10);
```

Figure 5.11: Johnny-five Code implementing from line 48 to line line 77

motor and servo motor and its pin for the Arduino board.Lines 64 through 77, the i2c configuration was done for the master board and the slave board. Lines 64 through 69, the i2c configuration was done for the first slave board, where the LED and fan motor will automatically turn on if the temperature rises above 39 degrees.
Lines 71 through 77, the i2c configuration was done for the again first slave board, where a different LED and fan motor will automatically turn on if the relative hu-

midity rises above 50 percent. Lines 79 through 86, the i2c configuration was for the

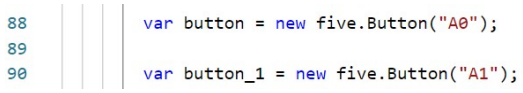


```
71  //i2c configuration for temp
72  var write_temp = (message) => {
73    this.i2cWrite(0x08, Array.from(message, c => c.charCodeAt(0)));
74  };
75  this.i2cConfig();
76  this.repl.inject({ write_temp });
77  console.log("Master and Slave boards configuration is initiated for temperature");
78
79  //i2c configuration for second slave
80
81  var write_vault = (message) => {
82    this.i2cWrite(0x08, Array.from(message, c => c.charCodeAt(0)));
83  };
84  this.i2cConfig();
85  this.repl.inject({ write_vault });
86  console.log("Master and Slave boards configuration is initiated for vault");
```

Figure 5.12: Johnny-five Code implementing from line 71 to line 86

second slave.

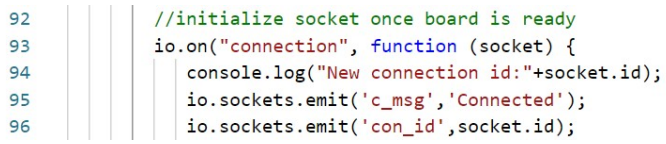
Lines 87–90, button and button1 instances created with analog pins A0 and A1.



```
88  var button = new five.Button("A0");
89
90  var button_1 = new five.Button("A1");
```

Figure 5.13: Johnny-five Code implementing from line 87 to line 90

Lines 92 through 96, the socket io is initialized and ready to use as soon as the



```
92  //initialize socket once board is ready
93  io.on("connection", function (socket) {
94      console.log("New connection id:"+socket.id);
95      io.sockets.emit('c_msg','Connected');
96      io.sockets.emit('con_id',socket.id);
```

Figure 5.14: Johnny-five Code implementing from line 92 to line 96

client-side is connected and displays the socket ID in the console and emits event 'c_msg' and 'con_id' to the client-side that the Arduino server is connected and shows the socket ID.

lines 99 through 135, initializing the status of the green, yellow, red and blue, motor and servo pins from Johnny Five modules, updating the current status of pins 13, 12, 8, 7, 10 and 3 on the console, and sending events on the client side.

Lines 142 through 145, listening for the client-msg event and printing the msg parameter to the LCD and console.

lines 148 to 153, listening for the 'client-msg1' event, with this parameter msg , will

```
98  //sending green led state upon new connection
99  myPin.query(function(state){
100   var led_state=state.state;
101   console.log("Broadcasting Pin 13 state:"+state.state);
102   socket.emit("led-state", led_state);
103 });
```

Figure 5.15: Johnny-five Code implementing from line 99 to line 103

```
142 socket.on("client-msg", function(msg){
143 lcd.clear().print(msg);
144 console.log("Message received msg:"+msg);
145 });
146
147 // Turning OFF Locker processing
148 socket.on("client-msg1", function(msg){
149
150 //lcd.clear().print(msg);
151 //console.log("Message received msg:"+msg);
152 write_vault(msg);
153 });
```

Figure 5.16: Johnny-five Code implementing from line 142 to line 153

transmit this message from the master board to the slave board to disable the Locker. lines 157 through 168, declaring and initializing a temp variable for the TH02 sen-

```
156 //setup for TH02 sensor
157 var temp = new five.Thermometer({
158   controller: "TH02",
159 });
160 //temperature function
161 temp.on("change", ()=>{
162     const {celsius, fahrenheit, kelvin} = temp;
163     console.clear();
164     console.log("Temp in Celsius " + celsius);
165     io.sockets.emit('temperature', celsius);
166     lcd.clear().print('temperature in Cel:'+ celsius);
167     write_temp(celsius+"*C");
168 });
169
170 var hygro = new five.Hygrometer({
171   controller: "TH02",
172 });
173 //temperature function
174 hygro.on("change", function(){
175   console.clear();
176   console.log("Relative Humidity " + this.relativeHumidity);
177   io.sockets.emit('hygrometer',this.relativeHumidity);
178   lcd.clear().print('relative humidity:'+ this.relativeHumidity);
179   write(this.relativeHumidity+"%RH");
180 });
```

Figure 5.17: Johnny-five Code implementing from line 156 to line 180

sor and outputting the temperature in degrees Celsius to the LCD and console, and emitting 'temperture' event with temperature values to the client-side.
lines 170 through 180, declaring and initializing a hygro variable for the TH02 sensor and printing the relative humidity as a percentage to the LCD and console, sending hygrometer event to the client-side, and sending a instruction from the master board

to slave board1 for slave action. lines 182 through 193 if the button was pressed

```
182   button.on("release", function() {
183
184     lcd.clear().print('Enter the 4 Digit Password');
185     write_vault("HIGH");
186
187   });
188
189   button_1.on("release", function() {
190
191     write_vault("LOW");
192
193   });
```

Figure 5.18: Johnny-five Code implementing from line 182 to line 193

and then released from the simulator. Then the button function will work with this "Release" event, it will print "Enter 4-digit password" on the LCD and instruct the "HIGH" message from the mainboard to the slave board 2 to act on the slave board side.

Likewise, pressing and releasing another function line 189 to 193 will command a "LOW" message from the master board to slave board 2 for slave side action.

Line 197 through 278 ,listening the state with event 'led-toggle'

```
196   //green led toggle function
197   socket.on("led-toggle", function(state){
198   //jhonny-five code here
199     if(state){
200       myLed.on();
201       lcd.clear().print("Green LED ON");
202       write("Green Led On");
203     }
204     else {
205       myLed.off();
206       lcd.clear().print("Green LED OFF");
207       write("Green Led OFF");
208     }
209     console.log("GREEN LED "+state);
210     myPin.query(function(state){
211       console.log("Pin 13 state:"+state.state);
212       socket.broadcast.emit("led-state", state.state);
213       socket.emit("led-state",state.state);
214     });
215   });
```

Figure 5.19: Johnny-five Code implementing from line 197 to line 215

if state = 1, turn on the green LED for pin 13, indicating a high value or 1, and print the green LED on the LCD and console.

else if state = 0, turn off the green LED for pin 13, indicating a low value or 0, and print the green LED off on the LCD and console.

And emitting and broadcasting of an event with updated states of pin 13 to the client side.

Likewise, we do the same with the events 'led-toggle2', 'led-toggle3', 'led-toggle4' to turn the red, yellow, and blue LED on and off.

```
280   // Fan toggle function
281   socket.on("fan-toggle5", function(state){
282     //jhonny-five code here
283     if(state) {
284       motor.forward(120);
285       lcd.clear().print("Fan ON");
286       write("FAN ON");
287     }
288     else
289     {
290       motor.stop();
291       lcd.clear().print("Fan OFF");
292       write("Fan OFF");
293     }
294     console.log("FAN "+state);
295     myPin5.query(function(state){
296       console.log("Pin 3 state:"+state.state);
297       socket.broadcast.emit("fan-state", state.state);
298       socket.emit("fan-state",state.state);
299     });
300   });
```

Figure 5.20: Johnny-five Code implementing from line 281 to line 300

line 281 through 280, listening the state with event 'fan-toggle'
if state = 255 , then rotate the motor with 120 rpm speed and print "FAN ON" on the LCD and the console.
else if state = 0, then stop rotating the motor, print "FAN OFF" on the LCD and the console.
And emitting and broadcasting of an event with updated states of pin 3 to the client side.

```
302   socket.on("servo-toggle", function(state){
303     //jhonny-five code here
304     if(state)
305     {
306       //servo lock
307       servo.to(180);
308       lcd.clear().print("Door Close");
309       console.log("Servo state : Lock");
310       write("Door Close");
311     }
312     else
313     {
314       //servo unlock
315       servo.stop();
316       lcd.clear().print("Door Open");
317       console.log("Servo state : unlock");
318       write("Door Open");
319     }
320     console.log("Servo "+state);
321     myPin6.query(function(state){
322       console.log("Pin 10 state:"+state.state);
323       socket.broadcast.emit("servo--state", state.state);
324       socket.emit("servo--state",state.state);
325     });
326   });
```

Figure 5.21: Johnny-five Code implementing from line 302 to line 326

line 301 through 326 ,listening the state with event 'servo–motor'
if state = 1500 , then rotate the servo motor in 180 angle. and print "Door Close" on the LCD and servo state on the console.
else stop the servo motor then print "door open" on the LCD and servo state on the console. And emitting and broadcasting of an event with updated states of pin 10 to the client side.

```
#include <Wire.h>
#include <LiquidCrystal.h>
const int maxlength = 64;
char buffer[maxlength];
char printable[maxlength];
const int motor =6;
const int motor2 =8;
const int LED1 =13;
const int LED2 =9;
int received = 0;

//initialize the library with the numbers of the interface
pins
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);
//String msg = "Hello";

void setup() {
  Serial.begin(57600);
  Wire.begin(8);                // join i2c bus with
address #8
  Wire.onReceive(receiveEvent); // register event
  pinMode(motor,OUTPUT);
  pinMode(motor2,OUTPUT);
  pinMode(LED1,OUTPUT);
  pinMode(LED2,OUTPUT);
  //set up the LCD's number of columns and rows:
    lcd.begin(16,2);
  //print a message to the LCD.
  //String message = String(loop1());
  //lcd.print(message);
             // start serial for output
}
//-------------------------------------------------------sla
```

Figure 5.22: Slave board 1 code of first part using Arduino IDE

From figure 5.22 to 5.30 ,this code is written in C ++ for slave board 1 and slave board 2. This code is written and compiled with the Arduino IDE.

```
ve board
Sketch-----------------------------------------------------
void loop() {
    if (received > 0) {
    memcpy(printable, buffer, maxlength);

    for (int i = 0; i < received; i++) {
      Serial.print(printable[i]);
    }
    Serial.println("");
    received = 0;
  }
  String message = String(printable);
  //Serial.println(message);
  int signal_1 = message.toInt();
  Serial.println(signal_1);
  String signal_2 = String(signal_1)+"*C";
  Serial.println(signal_2);
  Serial.println(message);
  //Serial.println(signal_2.charAt(2,4));
  if(signal_2 == message)
  {
    if(signal_1 > 38)
    {
      digitalWrite(motor,HIGH);
      //digitalWrite(motor2,HIGH);
      digitalWrite(LED1,HIGH);
    }
    else if(signal_1 < 39)
    {
      digitalWrite(motor,LOW);
      //digitalWrite(motor2,LOW);
      digitalWrite(LED1,LOW);
```

Figure 5.23: Slave board 1 code of second part using Arduino IDE

```
    }
  }

  String signal_3 = String(signal_1)+"%RH";
  Serial.println(signal_3);
  if(signal_3 == message)
  {
    if(signal_1 > 49)
    {
      //digitalWrite(motor,HIGH);
      digitalWrite(motor2,HIGH);
      digitalWrite(LED2,HIGH);
    }
    else if(signal_1 < 50)
    {
      //digitalWrite(motor,LOW);
      digitalWrite(motor2,LOW);
      digitalWrite(LED2,LOW);
    }
 }
lcd.clear();
lcd.setCursor(0,0);
lcd.print(message);
delay(250);
lcd.clear();
// function that executes whenever data is received from
master
// this function is registered as an event, see setup()
void receiveEvent(int howMany) {
  received = howMany;
  memset(buffer, 0, maxlength);

  for (int i = 0; i < howMany; i++) {
```

Figure 5.24: Slave board 1 code of third part using Arduino IDE

```
    buffer[i] = Wire.read();
  }
}
```

Figure 5.25: Slave board 1 code of fourth part using Arduino IDE

```
#include <LiquidCrystal.h>
#include <Servo.h>
#include <Keypad.h>
#include <Wire.h>
const int led = 13;
int inPin = 9;
int val = 0;
const int maxlength = 64;

char buffer[maxlength];
char printable[maxlength];

int received = 0;
int pos = 0;
//int signal_1 = 0;
String signal_1 ="";

const byte ROWS = 4; //four rows
const byte COLS = 3; //three columns
char keys[ROWS][COLS] = {
  {'1','2','3'},
  {'4','5','6'},
  {'7','8','9'},
  {'#','0','*'}
};
byte rowPins[ROWS] = {8, 7, 2, 3}; //connect to the row
pinouts of the keypad
byte colPins[COLS] = {4, 5, 6}; //connect to the column
pinouts of the keypad
String msg;
Keypad keypad = Keypad( makeKeymap(keys), rowPins, colPins,
ROWS, COLS );
Servo ServoMotor;
```

Figure 5.26: Slave board 2 code of first part using Arduino IDE

```
Servo ServoMotor1;
LiquidCrystal lcd(12, 11, A0, A1, A2, A3);

void setup(){

  Serial.begin(57600);
  Wire.begin(8);                // join i2c bus with
address #8
  Wire.onReceive(receiveEvent); // register event
  pinMode(led, OUTPUT); //declare LED as output
  pinMode(inPin,INPUT); //declare pushbutton as input
  ServoMotor.attach(10);
  ServoMotor1.attach(13);
  lcd.begin(16,2);
}

void loop()
{
  //delay(1000);
  val = digitalRead(inPin);
  if (received > 0)
  {
          memcpy(printable, buffer, maxlength);

          for (int i = 0; i < received; i++)
          {
             Serial.print(printable[i]);
          }
          Serial.println("");
          received = 0;
  }
  String message = String(printable);
  back:
```

Figure 5.27: Slave board 2 code of second part using Arduino IDE

```
if(message == "HIGH")
{
  signal_1 = "HIGH";
}
else if(message == "LOW")
{
  signal_1 = "LOW";
}

char key = keypad.getKey();


 if(signal_1 == "HIGH")
 {
      if (key != NO_KEY)
      {
         //Serial.println(key);
         lcd.print("*");
         msg = msg+String(key);

              if (msg == "1234" && msg.length() == 4)
              {
                  ServoMotor1.write(0);
                  delay(250);
                  ServoMotor.write(0);
                  msg = "";
                  val = LOW;
                  //delay(500);
                  //lcd.clear();
                  //delay(500);
                  lcd.clear();
                  lcd.print("VaultUnlock");
                  delay(100);
```

Figure 5.28: Slave board 2 code of third part using Arduino IDE

```
                    lcd.clear();
                    lcd.print("PressPushButton To Lock");
                    delay(500);
                    lcd.clear();
                    //lcd.clear();
                }
                else if(msg.length() > 3)
                {
                  if(keypad.getKey() != "1")
                   {
                     if(keypad.getKey() != "2")
                      {
                        if(keypad.getKey() != "3")
                         {
                           if(keypad.getKey() != "4")
                            {
                                msg = "";
                                lcd.clear();
                                lcd.print("Wrong
Password");
                                delay(200);
                                lcd.clear();

                                goto back;
                            }
                         }
                      }
                   }

                }
        }

  }
```

Figure 5.29: Slave board 2 code of fourth part using Arduino IDE

```
  else if(signal_1 == "LOW")
  {
      ServoMotor.write(90);
      delay(250);
      ServoMotor1.write(90);
      //msg = "";
     // lcd.clear();
      //delay(500);
      lcd.print("PressPushButton To Access Keypad");
      delay(500);
      lcd.clear();
  }
}

// function that executes whenever data is received from
master
// this function is registered as an event, see setup()

void receiveEvent(int howMany) {
  received = howMany;
  memset(buffer, 0, maxlength);

  for (int i = 0; i < howMany; i++) {
    buffer[i] = Wire.read();
  }
}
```

Figure 5.30: Slave board 2 code of fifth part using Arduino IDE

Chapter 6

Execution and Results

In Execution, we will see the process of accessing the modeling using Proteus Design Suite, whether it satisfies the API functionality with the specified parameters.

6.1 Unit Testing

In this testing, we have divided the simulation into small modules.

- **Master Board**
- **Slave Board 1**
- **Slave Board 2**

1. **Master Board** - In this module,I have added variables in the code that are 4 LEDs like green , yellow , blue , red , 2 push button , dc-fan , and servo motors.
    - Test Case 1 : We need to make sure that all the variables mentioned above must match the components I added to the schematic design.After saving and running this module, I verified that all the variables are declared and initialized with the appropriate Arduino pins and can be displayed in the VS Code console window.
    - Test Case 2 : In this case, I checked if there is a bug in the code or not. Thus I have fixed some bugs like missing parentheses, missing semicolons, functions undefined.
    - Test Case 3 : In this case, I also checked the correctness of the specified boundary conditions.

2. Slave Board 1 and Slave Board 2 - In this module, I have checked through Arduino IDE.

    - Test Case 1 : We need to make sure that all the variables must match the components I added to the schematic design. After compiling, I made sure that all the variables are declared and initialized with the corresponding pins of the slave board and can be displayed in the Arduino IDE console window.
    - Test Case 2 : In this case, I also checked three sections - define, void setup, and void loop - all of them checked.

6.2 Integration Testing

In this testing, I completed all modules once and tested the functionality after completing testing of individual modules - master board, slave board 1, and slave board 2.

In figure 6.2 , we see an Arduino simulation in the Proeteus Design Suite. To run the simulation, first enter node server.js in the terminal or console of the VS Code. Then press the play button in Proteus Design Suite.
Then open the browser and enter it: http://localhost:8080. We see a web page and this web page is the page for the Arduino board. In figure 6.1 we see a switch for the fan, LEDs and message box for sending any text and a switch for the locker, if the locker is open, we can close this by pressing the shutdown button.

The monitor display shows the status of yellow led, green led, blue led, red led, fan and door status, as well as temperature and relative humidity.
Figure 6.2 shows a rise in temperature to 39 degrees Celsius, then a yellow LED and one motor-fan turns on. The relative humidity rises above 50 percent, the red LED turns on and the other fan motor turns on.

6.3 Results

S.No	Components	Pin no.	High	LOW
1	Yellow Led	Pin 8	1	0
2	Green Led	Pin 13	1	0
3	Red Led	Pin 12	1	0
4	Blue Led	Pin 7	1	0
5	Fan	Pin 3	125	0
6	Door Lock	Pin 10	1500	2400
7	Push Button1	Analog Pin A0	HIGH	LOW
8	Push Button2	Analog Pin A1	HIGH	LOW

Results of the component used in the simulation at high and low values

We obtained these values by printing the status of these pins and analog pins to the console or terminal during simulation.

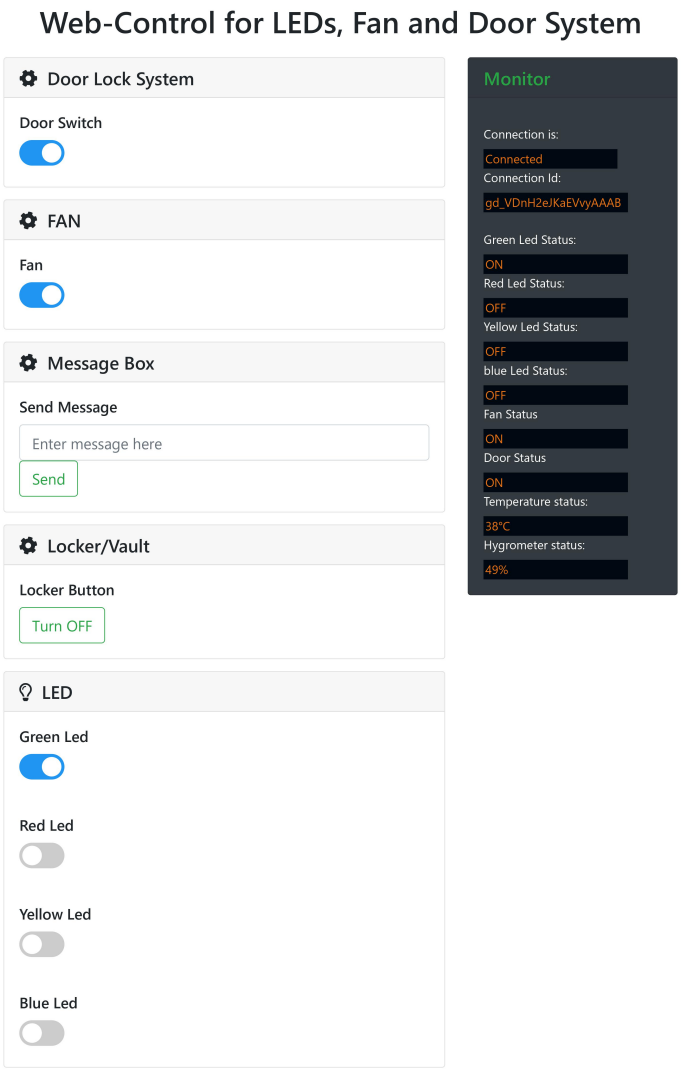


Figure 6.1: Webpage that successfully controls Arduino board using js sever node

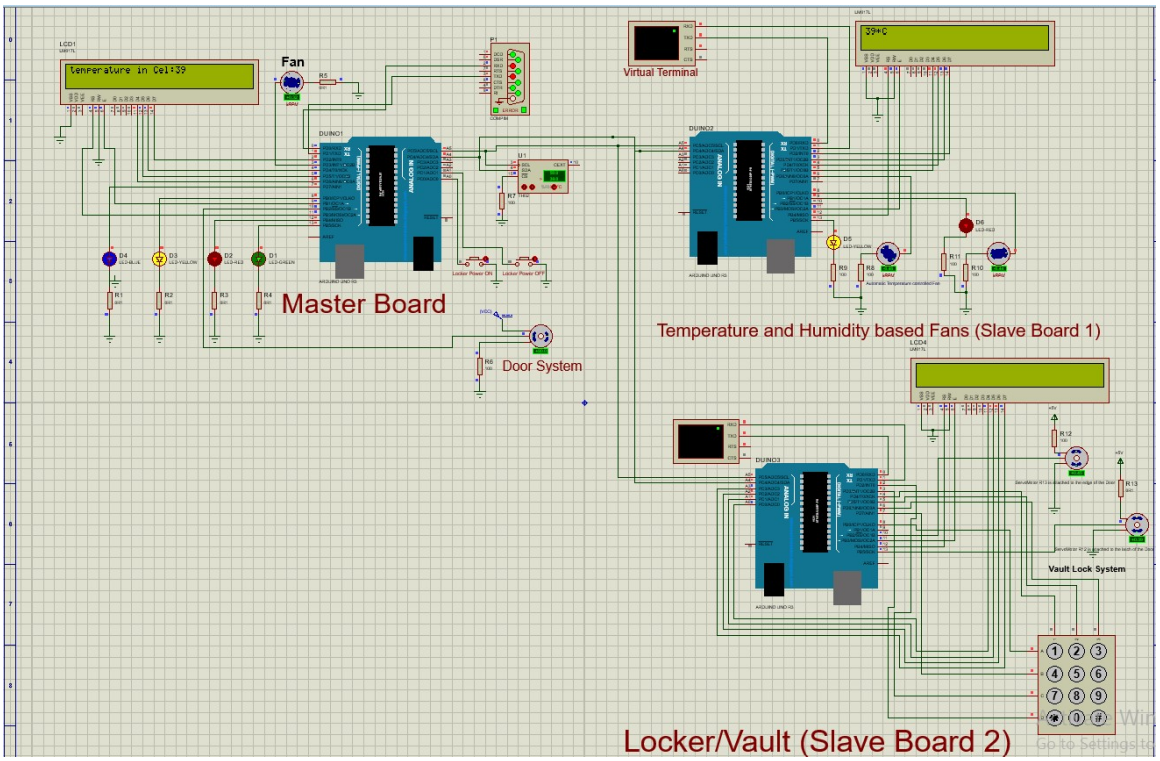


Figure 6.2: After completing all tests, the simulation was successful

Chapter 7

Task Analysis

The Action required to set up and run a project. Initially we need to create a schematic design using the Proetus Design Suite, make sure the connection is perfect as I designed.
We need to keep the files on our system in an easily accessible place as we will need them later.
Then we need to access the location where we saved our files. Make sure all files, program file, design file and API file should be together.
After writing the code, make sure that all files are compiled and all the required API libraries must be imported.
In the VS Code Terminal, enter node javascript_filename.js to run it. Make sure Proteus Design Suite is running.
After that, open any browser, enter http://local:8080, and the real work will start.
We will analyze several cases, I will help to understand the project thoroughly.

- Case 1 : Now the simulator is ready to perform the operation, we need to switch the LEDs like yellow, green, blue, red, fan and door. Thus, we can see that all LEDs, fan and door are working in Proteus Design Suite.

- Case 2 : In Proteus Design Suite, a temperature rise of more than 39 degrees Celsius through the TH02 sensors causes the fan and yellow LED to turn on. And an increase in RH of more than 50 percent through this sensor causes another fan to turn on and red led.

- Case 3 : On the main arduino board, pressing button 1 and button 2 turns on the keyboard, so it asks for a 4-digit password on the slave side.

- Case 4 : Pressing button 2 will turn off the keypad on slave board 2, it will not be possible to enter the password. We can also turn off locker from the browser side by clicking the turnoff button on the client side.

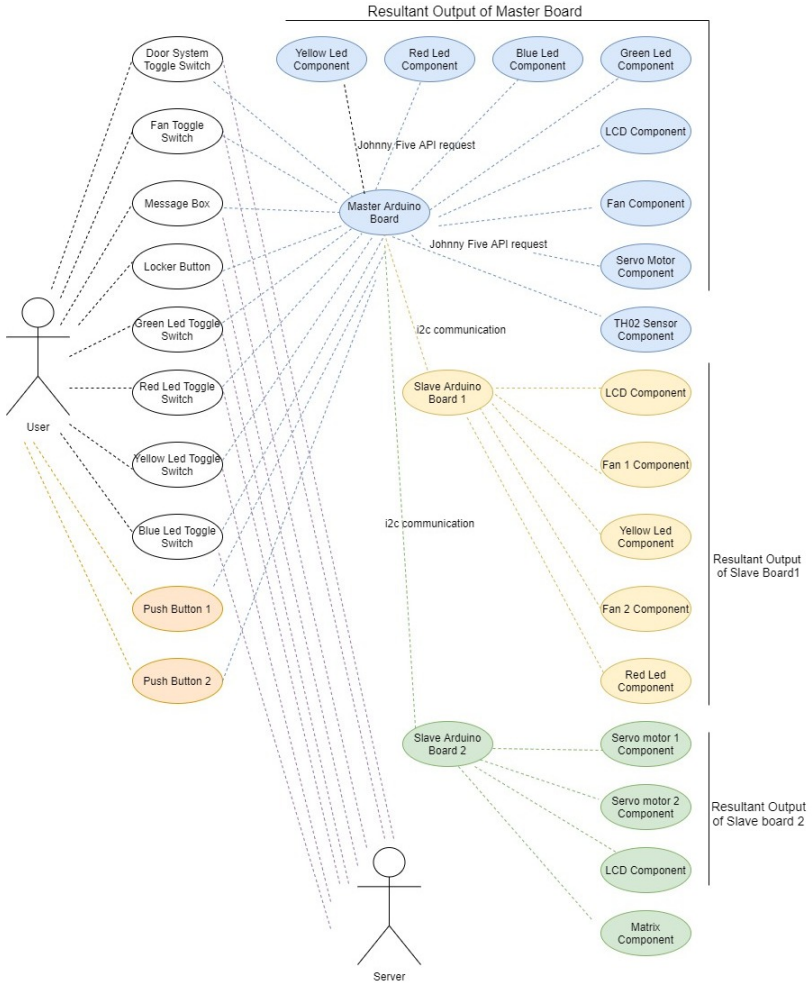


Figure 7.1: Use case diagram

Chapter 8

Conclusion and Future Work

8.1 Conclusion

This project is best for teaching a beginner to get started with Arduino through simulation. This project is beneficial to become an intermediate user from a beginner. The only thing we need to know is all those components and API library that will use it.
And we will also learn important APIs on Arduino, and the johnny-five library is really useful for writing Arduino code.

8.2 Future Work

- For learning purposes, there is no need to buy an Arduino board and its equipment. Students can learn along with it with simulation.
- This can be a blueprint for the embedded software engineer where they will first design and simulate and then they plan to buy and work with the actual Arduino board and its equipment.
- Perhaps I will be one of the students who will later build real projects, having received knowledge from this project.

ABSTRACT

In this document, we are exploring a working Arduino system using the API library. This will teach students how to learn it with the Johnny-Five library. This library will link the provided API library with Arduino and its components. Students can access Arduino and other devices without touching them using the simulator. After using the Johnny-Five library, students will have a basic knowledge of the Arduino Board Simulator and its features as it is free source code. They will have the opportunity to realize imaginative ideas and develop libraries.

Keywords: Arduino, Johnny-Five

Bibliography

[1] Lyza Danger Gardner,"JavaScript on Things Hacking hardware for web developers (2018)".

[2] Richardson, Matt,"Getting Started with BeagleBone: Linux-Powered Electronic Projects With Python and JavaScript (2013)".

[3] Rick Waldron,"JavaScript Robotics: Building NodeBots with Johnny-Five, Raspberry Pi, Arduino, and BeagleBone (2015)"

[4] Perch, Kassandra ,"Learning JavaScript Robotics (2015)".

[5] Monk, Simon,"Programming Arduino: Getting Started with Sketches, Second Edition (2016)"