

**A Project Report**

**On**

**IOT Based Smart Garage Door Opener**

Submitted by:

**Dawinder Kaur (C0765505)**

**Manpreet Kaur (C0757604)**

**Shivinder Kaur (C0751078)**

**Gopi Krishnan K. GopalaKrishnan (C0753811)**

**Raghav Chopra (C0772292)**

Under the guidance of

**Dr. Mike Aleshams**

**ESE-4009 Embedded System Design Project**

(January 2021 - April 2021 )

**Abstract**

This major project is all about the embedded systems and its applications in the real world with the help of beaglebone black and arduino software. As a result of enhanced civilization and modernization, human nature demands more comfort in their life. The person seeks ways to do things easily which saves time. So thus, the automatic gates are one of the inventions that brings comfort and ease in their daily life. Before as well people used garage doors to provide security to their vehicles but nowadays, in this modern world people need more security so that an unknown person will not access the garage door. That's why, IOT based garage doors are available now which secures the doors more accurately and people can save their entrance and exit times on the cloud along with the personal details of the vehicles. The owner will get the notification through the GSM module if any unknown person will try to open the garage door. The PIR sensor detects the vehicle at the time of opening the door and also finds the obstacle at the time of closing the door.

**Acknowledgements**

In our efforts towards the realization of the project work,we have drawn on the guidance of many people and we would like to express my heartfelt gratitude to all the people who helped in fulfilling the accomplishment of this project.

We would like to express my deep gratitude to Michael Vourakes, Director, Lambton College in Toronto for providing us an opportunity to study and in providing infrastructure and resources required to complete this project.

We also would like to extend our gratitude to Dr. Mike Aleshams for the unconditional support and coordination to make this project a success. We also would like to thank him for sparing time and providing necessary information patiently and providing efficient tips for debugging. His constant motivation and inspiration have led to successful completion of the project.

We also extend Dr. Takis Zourntos, Course Coordinator for Embedded System Design for critical advice and guidance without which the project would not have been possible. We also extend our gratitude to all other Professors of Lambton College in Toronto for their guidance and support in compiling the project and those people who have helped us knowingly and unknowingly for the completion of the project.

Lambton College in Toronto is a great place to study and work, largely because of supporting faculty members. Thanks to all my colleagues for supporting and encouraging our idea.

TABLE OF CONTENTS Page No.

**List of Figures**  5

**Chapter I - Introduction**  8

Overview 8

Problem statement 8

Goal and Objective 8

Scope of project 9

Outcomes and Benefits 12

Facilities and Resources 12

Procedure and Methodology 13

**Chapter II - Literature Review**  14

**Chapter III - Requirement and Analysis** 16

Hardware and Software 16

Hardware and Software Tools 17

Power Requirement 17

Block Diagram 18

**Chapter IV - Design**  21

Project Schematic 21

Knowing Hardware 22

Knowing Hardware and Software Tools 30

Cloud service - Thingspeak 34

PCB Design 36

**Chapter V - Implementation and Test**  39

Flow chart 39

Interfacing with BBB 43

Coding 66

Final Model 70

**Chapter VI - Evaluation**  72

Introduction 72

Minimum Requirements 72

Troubleshooting 72

**Chapter VII - Conclusion**  74

Future Work 74

**Chapter VIII - User’s Guide**  75

**References** 79

**List of Figures**

|  |  |
| --- | --- |
| Figure No. | Description |
| 2.1 | *Circuit Diagram of Existing Project* |
| 2.2 | *Block Diagram of Existing Project* |
| 3.1 | *Block Diagram of Final* |
| 4.1 | *Schematic Design of Project* |
| 4.2 | *Beaglebone Black* |
| 4.3 | *Pin Configuration of Beaglebone Black* |
| 4.4 | *Arduino UNO* |
| 4.5 | *IR Proximity Sensor* |
| 4.6 | *SIM800L Module GSM Module* |
| 4.7 | *Buzzer* |
| 4.8 | *4x4 Matrix Keypad* |
| 4.9 | *L298N DC Motor Driver* |
| 4.10 | *Limit Switches* |
| 4.11 | *20x4 LCD Display* |
| 4.12 | *ESP32 Wifi Module* |
| 4.13 | *Push Button* |
| 4.14 | *DC 6V Gear Motor* |
| 4.15 | *LED Lights* |
| 4.16 | *Logic Level Controller* |
| 4.17 | *Digital Multimeter* |
| 4.18 | *Breadboard* |
| 4.19 | *Jumper Wires* |
| 4.20 | *General Purpose Tools* |
| 4.21 | *Soldering Kit* |
| 4.22 | *Thingspeak channel created for project* |
| 4.23 | *Setting channel for read and write API* |
| 4.24 | *Visualization in Thingspeak* |
| 4.25 | *Easy EDA Introduction window* |
| 4.26 | *Circuit Schematic Drawing* |
| 4.27 | *Search window for components* |
| 4.28 | *PCB Design for Project* |
| 5.1 | *Flowchart using Webpage* |
| 5.2 | *Flowchart using LCD and Keypad* |
| 5.3 | *Interfacing BBB with IR Sensors* |
| 5.4 | *Interfacing ESP32 with BBB* |
| 5.5 | *Interfacing BBB with Motor Driver L298N and DC Motor* |
| 5.6 | *Interfacing BBB with Limit switches* |
| 5.7 | *Interfacing BBB with Buzzer* |
| 5.8 | *Interfacing BBB with Push Button* |
| 5.9 | *Interfacing BBB with Arduino* |
| 5.10 | *Interfacing Arduino with LCD Display* |
| 5.11 | *Interfacing Arduino with Keypad* |
| 5.12 | *Interfacing Arduino with GSM Module* |
| 5.13 | *Connections of Hardware Components* |
| 5.14 | *Backview of the project* |
| 5.15 | *Final model of our project* |
| 8.1 | *Beaglebone with Adapter* |
| 8.2(a) | *Uploading Code to the Wifi Module* |
| 8.2(b) | *Connecting Wifi to the Wifi Module* |
| 8.3(c) | *Wifi getting connected* |
| 8.3(d) | *ESP32 Web Server* |
| 8.4 | *Keypad and LCD* |

**Chapter I**

**Introduction**

**Overview**

This project is the modification and upgradation in the existing project of Smart garage door opener. In this project, we have used a GSM module, Beaglebone black, Arduino, Cloud service and other hardware components. The purpose of this project is to provide more security to the garage doors and also we have used one cloud service i.e., Thingspeak which saves the details of the vehicle and the status of opening and closing the door.

It has one most important feature that is also related to security, if someone tries to access the garage door by entering the password multiple times, then a notification will be sent to the owner of the garage in order to avoid any mishappening.

**Problem Statement**

The door plays an important role in home security. So, providing a secured garage door system for houses has become necessary. This project deals with opening and closing of garage doors by giving commands through webpage. The principle of operation behind the working of this project lies in the functioning of NodeMCU ESP12E. It works as a wifi module that provides access to the webpage through a local web server. The webpage displays one button to open and close the garage door using a relay module and DC motor. It has some limitations like the microcontroller NodeMCU ESP12E can only be used as a wifi module and also it is not an IOT based project. It has no security features. Because of all these limitations, we have decided to modify the existing project with new features like cloud service and additional security features.

**Goal and Objectives**

**Goal**

Make an IOT based smart garage system with GSM and cloud based technology running with Linux kernel by the end of April 2021.

**Objective**

* Connecting a host processor Beaglebone black to one of the available cloud services like thingspeak, Aws and so on.
* Interface the IR sensor which will help us to know if any obstacle blocks the garage door at the time of opening and closing.
* Configuring the wifi capabilities to connect cloud service.
* Send a notification if an unknown person tries to access the door by entering the wrong password three times.

**Scope of the project**

**Deliverable**

* This project will access the garage door via webpage as well as keypad and LCD for opening and closing.
* Transfer all the data to the cloud service.
* Generate a PCB design for the project.
* Intimate about the door activity via SMS services through cloud services.
* Plug and Play the project that is the smart garage door should function without the help of any other device or laptop to execute the codes written and get the exact output.

**Milestones**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **S.no.** | **Task Title** | **Start Date** | **End Date** | **Person In-charge** |
| 1 | Project Proposal | 27-jan-2021 | 12-feb-2021 |  |
| 2 | Deciding and ordering hardware components | 13-feb-2021 | 15-feb-2021 | Gopi Krishnan |
| 3 | Testing hardware components Part 1 | 16-feb-2021 | 22-feb-2021 | Raghav Chopra |
| 4 | Testing hardware components Part 2 | 16-feb-2021 | 22-feb-2021 | Manpreet Kaur |
| 5 | Designing Schematic Diagram | 23-feb-2021 | 26-feb-2021 | Shivinder Kaur |
| 6 | Interfacing BBB with IR sensor (to detect the status of garage) | 27-feb-2021 | 5-march-2021 | Dawinder Kaur |
| 7 | Interfacing WIFI Module ESP32 with BBB | 27-feb-2021 | 5-march-2021 | Shivinder Kaur |
| 8 | HTML coding for web page | 6-march-2021 | 10-march-2021 | Dawinder Kaur |
| 9 | Interfacing BBB with Motor Driver L298N and connecting it to DC motor | 11-march-2021 | 15-march-2021 | Gopi Krishnan |
| 10 | Interfacing of Limit Switch with BBB | 15-march-2021 | 19-march-2021 | Raghav Chopra |
| 11 | Interfacing BBB with IR sensor (to detect obstacle) | 15-march-2021 | 19-march-2021 | Manpreet Kaur |
| 12 | Interfacing Buzzer with BBB | 20-march-2021 | 26-march-2021 | Raghav Chopra |
| 13 | Interfacing Button with BBB | 23-march-2021 | 28-march-2021 | Shivinder Kaur |
| 14 | Interfacing Cloud services with ESP32 | 28-march-2021 | 2-april-2021 | Dawinder Kaur |
| 15 | Interfacing Arduino with BBB | 28-march-2021 | 2-april-2011 | Gopi krishnan |
| 16 | Interfacing LCD display with arduino | 3-april-2021 | 9-april-2021 | Manpreet Kaur |
| 17 | Connecting Keypad with arduino | 3-april-2021 | 9-april-2021 | Raghav Chopra |
| 18 | Interfacing Arduino with GSM Module | 10-april-2021 | 16-april-2021 | Gopi Krishnan |
| 19 | PCB design using particular software | 10-april-2021 | 16-april-2021 | Manpreet Kaur |
| 20 | Zero PCB Implementation Part 1 | 17-april-2021 | 23-april-2021 | Shivinder Kaur |
| 21 | Zero PCB implementation  Part 2 | 17-april-2021 | 23-april-2021 | Dawinder Kaur |
| 22 | Final Report | 24-april-2021 | 25-april-2021 |  |
| 23 | Final Presentation | 27-april-2021 | 28-april-2021 |  |

**Limitations**

* When the user enters the wrong password to open the garage door more than 3 times then the owner will get notification but the LCD and keypad does not get locked. The user still can access them.
* If the owner forgets the password then there is no option to reset the password directly using keypad and LCD.

**Outcomes and Benefits**

* This project is powered up with the power adapter so no need to worry about the power unless there is no electricity in the house.
* If due to some reason the owner will not be able to open the door by using a keypad then he/she can access it by using cloud service.
* It could be an upgrade to the existing project.
* It will help to save the details of the door status at the time of opening and closing to the cloud service which the owner can review later on.

**Facilities and Resources**

**Laboratory**

* As we have our classes online so we were not able to access the Embedded Systems Lab of our college. So we created a miniature laboratory at our own places to do all the work on the project.
* Soldering stations, Digital Multimeters, Linux or PuTTy installed PCs and all the other required components were available at our homemade labs.

**Intellectual Resources**

* Data sheets of GSM module, Beaglebone black, Arduino and other hardware components.
* Forums of Beaglebone, Element14 and components101 communities.
* Exploring Beaglebone tools and techniques for building with Embedded Linux (Second Edition) from Derek Molloy.

**Procedure and Methodology**

* Getting the hardware requirements ready and ordered for the project sideways testing the components and understanding their working as they keep on arriving.
* Install and update the devices such as the Beaglebone with the latest image OS in order to get to use some of the latest features available in terms of device trees and drivers.
* Upon the getting the testing results positive and approved move and design the circuit schematic using software like EasyEDA, Fritzing, KiCad or Eagle, etc.
* Installation of various libraries which will make our task of controlling the pins of Beaglebone easy and communication between peripherals and sensor simpler.
* Keeping the schematic in mind gets the connection ready for the interfacing in part as this will help us to know in case we go wrong or if the system is not working at some points.
* We start with the interfacing of BBB with IR to detect the status of the garage.
* To make the data available on the cloud service we need to establish a wired connection between ESP 32 and BBB and send the appropriate data to the cloud and make it visible to the user.
* After that, interface all the hardware components with each other.
* At the end of all the interfacing we will integrate them and check if we get the desired functionality as mentioned in the proposal.
* In order to minimize the amount of wiring we must generate the PCB design. The PCB could be designed with the same software that we may have used in making the schematic on the software in step iii).
* Solder the devices on the PCB and test for the final demonstration of the project.
* Make the project run in such a way that when powered on the program should auto-run without the help of any sort of connection with laptop or any external devices.

**Chapter II**

**Literature Review**

**Wifi Based Smart Garage Door Opener**

This project deals with opening and closing of garage doors by giving commands through webpage. The principle of operation behind the working of this project lies in the functioning of NodeMCU ESP12E. It works as a wifi module that provides access to the webpage through a local web server. The webpage displays one button to open and close the garage door using a relay module and DC motor.

**Circuit Diagram:**

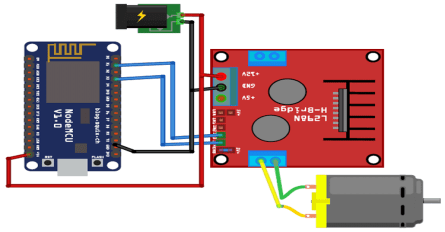
****

Fig.2.1 *Circuit Diagram of Existing Project*

**Block Diagram:**

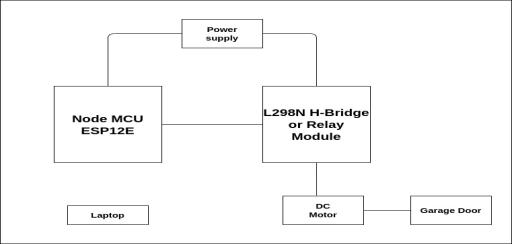


Fig.2.2 *Block Diagram of Existing Project*

**Working:**

The working principle of the project is giving commands through a web page displaying a message about garage door “open and close” status. So when the garage door is closed then the message is displayed as “The Garage Door is closed” and the OPEN button is shown on the screen. The Node MCU ESP12E is programmed accordingly using an Arduino IDE by providing a wifi username and password in code. When the user opens the webpage using the IP address from the serial monitor window the web page opens. When the user clicks on the open or close button the message goes to Node MCU which passes the command to the relay module to rotate the DC motor “clockwise or anticlockwise” which helps in opening and closing the garage door.

**CHAPTER III**

**Requirement and Analysis**

**Hardware and Software**

In this project, we are using hardware and software components along with the hardware and software tools so that we will get the proper output. Below, we have attached the list of hardware and software components and also the tools which we have used in our project.

**Hardware components :**

1. Arduino Uno
2. Beaglebone Black
3. ESP32 wifi module
4. 4x4 matrix keypad
5. 20x4 LCD Display
6. SIM800L Module GSM/GPRS
7. Power Supply
8. IR proximity sensors
9. Buzzer Electronic alarm
10. Push button
11. MicroLimit switch
12. DC Motor
13. L298N Motor Driver
14. SanDisk SD card 16 gb
15. Logic level controllers
16. PCB Board
17. 5mm LED light assorted kit

**Software components:**

1. Latest image of debian for beaglebone black
2. Amazon Web Services or ThingSpeak or any other free IoT service provider.
3. C/C++ Programing
4. HTML programming

**Hardware and Software Tools**

**Hardware Tools:**

1. Multimeter
2. Breadboard
3. Jumper Wires/ Silicon hook up wires
4. General purpose tools
5. Soldering Kit
6. Laptop or host machine

**Software Tools:**

1. EasyEDA for PCB and schematic design
2. Eclipse IDE for programming
3. PuTTy software
4. Compiler
5. Debugger
6. Assembler
7. Arduino IDE

**Power Requirement**

We are providing a sufficient amount of power to get the system working and performing all the tasks. Thus, looking at the requirement for our project we find that we have two major power consuming components which are:

1. Beaglebone Black – 2 A @ 5V

2. Arduino based GSM Shield – 2 A @ 5V

So, estimated power requirements for full product assembly will be approximately 4V @ 4A for smooth and interrupt free operation. As most of the hardware components that we want to use for the project and when we look at the pin out diagram of the same, we realise that we can power them up directly from the Beaglebone thus no extra power supply is required for them.

**Block Diagram**

For the final solution, we will implement the project as we have discussed in the second solution but with some modifications. We will add a GSM module in the project which will be connected with Arduino. This module will be used for sending notifications to the user via SMS in case the door remains open. Also, the keypad will be used to enter a password to open the door but for closing the door we will just need to press enter key.

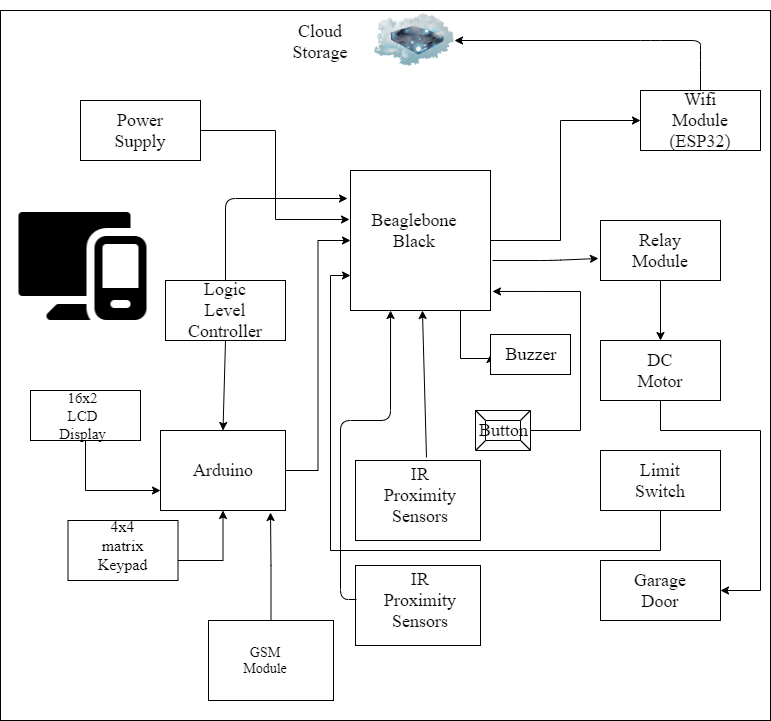


Fig 3.1 *Block diagram for the final solution*

**Features:**

* Using Beaglebone as the main microcontroller which has high processing speed and more advanced features.
* Using Arduino as a slave device which is supporting the alternate solution of opening and closing the door.
* The project is IoT based as it is connected to Wifi for cloud storage.
* IR Proximity Sensors will be used to detect any object in between the garage door while closing.
* Use of a buzzer if the door remains open for a long time.
* Also a GSM module has been added which will notify the owner via SMS that the door remained open.

**Chapter IV**

**Design**

**Project Schematic**

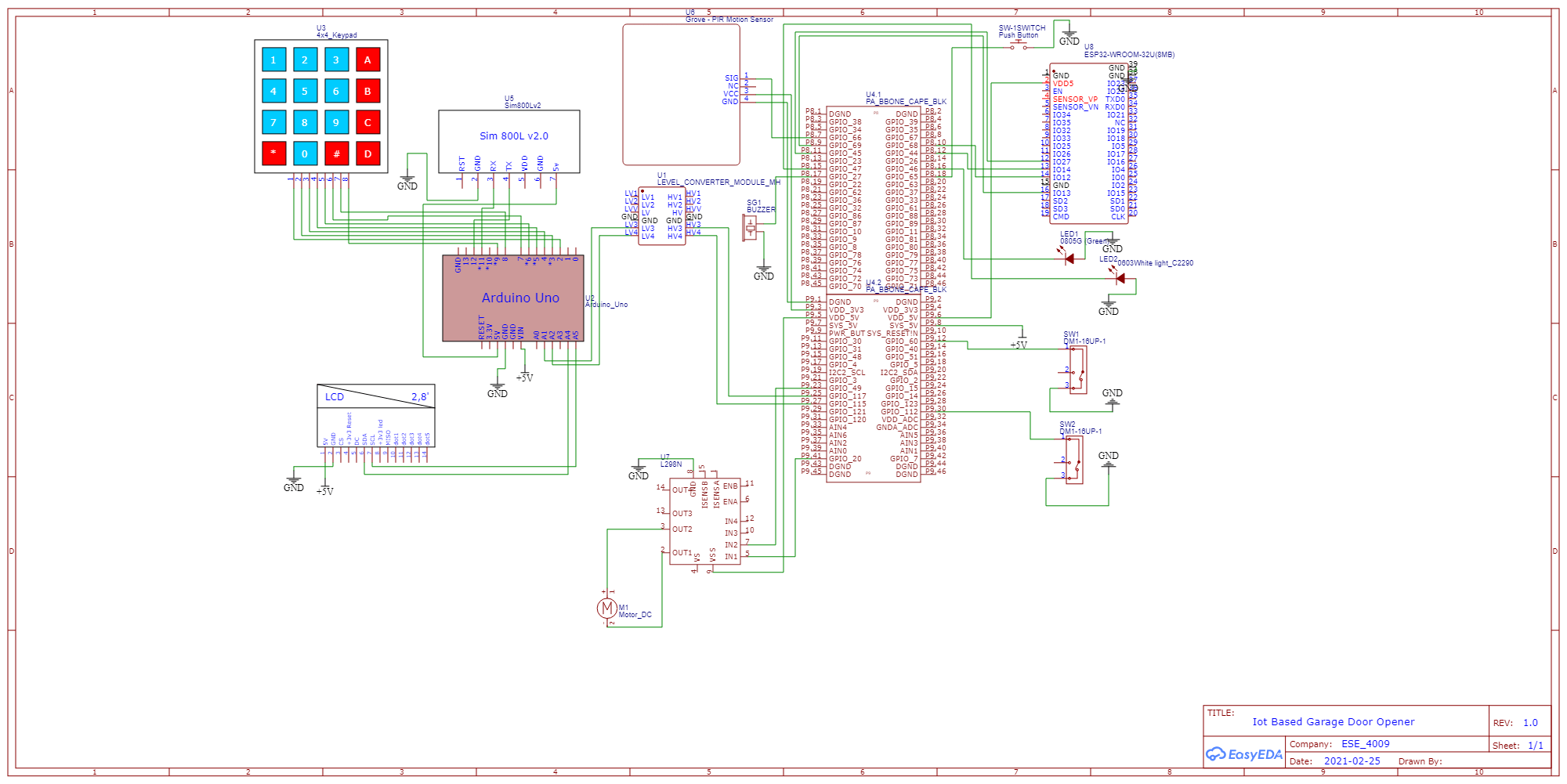


Fig 4.1. *Schematic Design for the project*

Above figure shows the schematic capture for the project which will help us wire the same. To draw the schematic, we use EasyEDA software. EasyEDA is a free open-source platform and is best suited for beginners in schematic capture. As we can see in this schematic how the hardware components get connected with the Beaglebone black. All Beaglebone Boards have the same pin configuration. At the later stage of the project, we used this schematic and converted it into the PCB Layout which we can use to get PCB manufactured for the project. Before moving on further let us first discuss and know the hardware in detail which will help us in programming the system and reason for our connection.

**Knowing Hardware**

**Beaglebone Black**

The Beaglebone black is a low cost, open hardware and expandable computer launched by a community of developers from Texas Instruments. It consists of a large amount of input and output pins and also it has on board interfaces which enables the devices to connect smoothly with the beaglebone black.



Fig 4.2. *Beaglebone Black*

Beaglebone Black has so many technical specifications and also it has a large number of GPIO I/O pins. It is an open source hardware technology which gives options to the manufacturer to integrate ARM technology for developing cloned beaglebone development boards and also it provides options to install a wide range of android and linux based operating systems.

**Pin configuration of Beaglebone Black:**

It has two headers P8 and P9 and both headers have 46 pins. Also, it has GPIO pins as well which we can use at the time of interfacing with other hardware components.

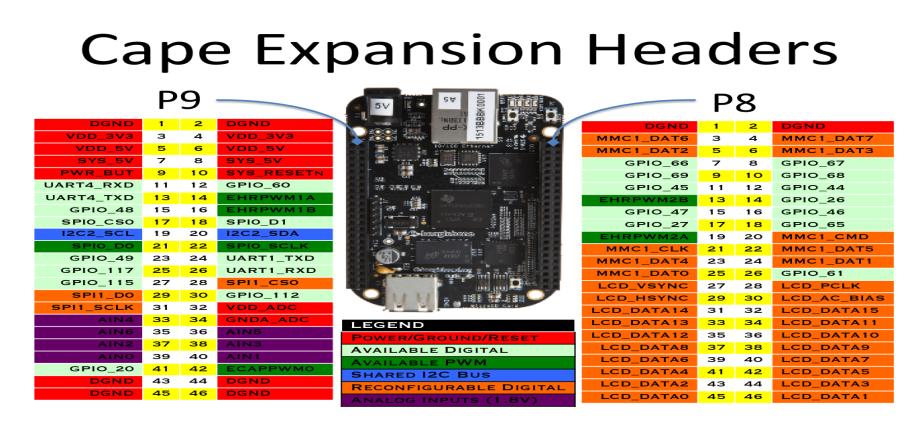
****

Fig 4.3. *Pin Configuration of Beaglebone Black*

**Arduino Uno**

The Arduino UNO is an open source microcontroller board which is based on the microchip ATmega328P (for Arduino UNO R3) or microchip ATmega4809 (for Arduino UNO wifi R2) microcontroller by Atmel and was the first USB powered board developed by Arduino.It contains everything needed to support the microcontroller, simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started.



Fig 4.4. *Arduino UNO*

**IR Proximity Sensors:**

In our project we are using IR sensor EK1254 which is an obstacle detecting sensor. We will place this sensor on the garage door to avoid any collision while the door will be closing. This module can be directly connected to beaglebone black. It can be connected to both 5V and 3.3 V microcontrollers.

The module does not require to contact the object to detect it; it detects within the distance between 2-30 cm. They are capable of detecting motion in presence or absence of the light almost with the same reliability.

The figure shows the IR sensor. It has a transmitter and receiver to detect any obstacle.



Fig 4.5. IR Proximity Sensors

**SIM800L Module GSM GPRS**

In our project we want to notify users via SMS, if the garage door remains open for a long time and sends an alert message. For this purpose we will use the SIM800L GSM module. It is a small miniature cellular module which allows for GPRS transmission, sending and receiving messages. Here are the specifications of this module:

* Quad-Band GSM 850/900/1800 / 1900MHz
* GPRS multi-slot class 12/10
* GPRS mobile station class B
* GSM 2/2 +FM frequency between 76 MHz and 109 kHz adjustable by step 50AT command set support (3GPP TS 27.007, 27.005 and augmented by SIMCOM AT)
* Class 4 (2W at 850 / 900MHz)
* Class 1 (1 W at 1800 / 1900MHz)
* Serial interface
* SIM card interface 3 V / 1.8 V

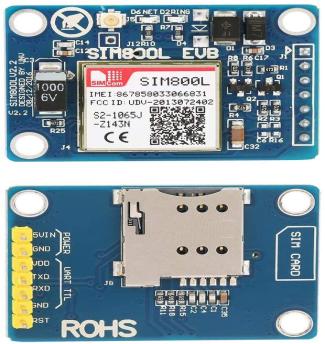


Fig 4.6. *SIM800L Module GSM GPRS*

**Buzzer**

To alert the user about obstacles we need an actuator , for this purpose we are using a buzzer and this will be connected to beaglebone black. For our project we are using Cylewet CLT1036. It requires <25mA and 4-8 voltage to operate.



Fig 4.7. *Buzzer*

**4X4 Matrix Keypad**

Matrix keypads are the kind of keypads you see on cell phones, calculators, microwaves ovens, door locks, etc. Membrane keypads are made of a thin, flexible membrane material. They do come in many sizes 4×3, 4×4, 4×1 etc. Regardless of their size, they all work in the same way.Let’s take 4×4 keypad as an example. It has a total of 16 keys. Beneath each key is a special membrane switchAll these membrane switches are connected to each other with a conductive trace underneath the pad forming a matrix of 4×4 grid.

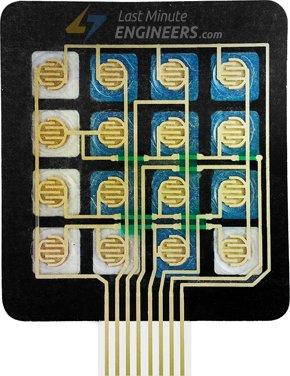


Fig.4.8. *4X4 Matrix Keypad*

**L298N DC Motor Driver**

L298N Motor Driver interface with beaglebone black which controls both speed and spinning direction of two DC motors.

This can be achieved by combining these two techniques.

* PWM (Pulse Width Modulation) – For controlling speed
* H-Bridge – For controlling rotation direction



Fig.4.9 *L298N DC Motor Driver*

**Limit Switches**

The micro limit switch, or micro switch, is a limit switch commonly found on control circuits. In our project we will use these switches to sense motion and to open and close the door with the actuators.In our project we will use PChero part number iS008 switch.

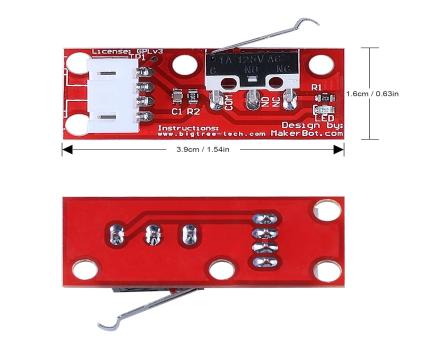


Fig.4.10 *Limit Switches*

**20X4 LCD Display**

In our project we will notify the user via lcd display whenever the user will enter the wrong password. To display certain messages to notify the user for access granted or access denied. For our project we are using 20\*4 LCD display module. It can show 20 characters in one row which is perfect to display a small message to the user. This specific product includes a serial interface adapter and therefore no need to buy an extra adapter. Less I/O ports are occupied, only four - VCC, GND, SDA (serial data line), SCL (serial clock line). and thus no connections are required.



Fig.4.11 *20X4 LCD Display*

**ESP32 Wifi Module**

As we have discussed that our project is going to be IOT based, for that we need internet connection. As we are using beaglebone black, which doesn’t have any internal wifi module. Therefore, we will use a separate Wifi Module for internet connectivity. For this purpose we are using the ESP32 wifi module. This module can be used for low power sensors networks. ESP32 also integrates a wealth of peripherals, including capacitive touch sensors, Hall sensors, low noise sensor, SD card interface, Ethernet interface, high-speed SDIO / SPI, UART, I2S and I2C.



Fig.4.12 *ESP32 Wifi Module*

**Push Button**

In our day to day life we encounter various push button switches such as on medical equipment, automated production lines, and communication equipment. It is a very simple component, but because it controls the entire control circuit, it is also known as “**Control Switch”**.

The push button is usually used to turn on and off the control circuit, and it is a kind of control switch appliance that is widely used. It uses electrical automatic control circuits to manually send control signals to control contactors, relays, electromagnetic starters, etc.



Fig.4.13 *Push Button*

**DC 6V Gear Motor**

DC gear motor has an integrated M3\*55mm threaded rod as its output shaft. It has a series of gears, which in turn creates more torque. The range of driver voltage between 5 to 35V and driver current is 2A. Its maximum motor supply current is 2A and maximum power is 25W. The revolutions per minute of the DC motor is 150RPM.



Fig.4.14 *DC 6V Gear Motor*

**5mm LED Light**

In this project, we are using LEDs for various purposes like to show the status of opening and closing the garage door and also we have LEDs which are connected to the IR proximity sensor in order to detect any obstacle.



Fig.4.15 *5mm LED Lights*

**Logic Level Controllers**

In our project, we have used logic level controllers to shift the voltage level between Beaglebone Black and Arduino because as we know Beaglebone Black needs 3.3V and Arduino operates at 5V. This controller converts the 3.3V to 5V so that Beaglebone and Arduino can communicate smoothly.

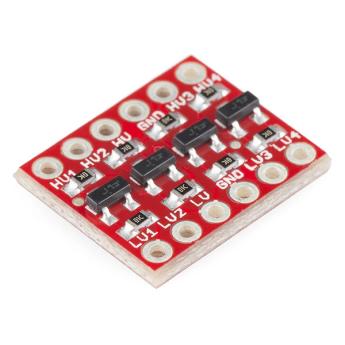


Fig.4.16 *Logic Level Controller*

**Knowing Hardware and Software Tools**

**Hardware Tools**

* **Multimeter :** It is one of the most important measurement and testing devices in any project. It is used popularly for keeping a track of current, voltage, resistance, and other values, all of which are very critical for the safe and balanced working of the project. It plays a big part in component testing as well. There are two types of multimeter in the market. First is analog multimeter and the second is digital multimeter, but in our project we are using digital multimeter because of its better accuracy.



Fig 4.17 *Multimeter*

* **Breadboard:** It is the base of any project. It is popularly used for making a model to check the proper functioning during design before soldering the components.



Fig4.18 *Breadboard*

* **Jumper Wires:** Classified as male to male, male to female and female to female, It is another prototype tool which is used for establishing connections on the breadboard for designing a prototype model.



Fig 4.19 *Jumper Wires*

* **General purpose tools:** For designing and modeling of the project many general purpose tools will be required. These tools are used during the project design and implementation. For ex: A wire cutter, wire stripper, ruler, measuring tape etc.



Fig 4.20 *General Purpose Tools*

* **Soldering Kit :** It is a tool used by hobbyists, students and engineers for establishing permanent connections between various components. In this project it is used for soldering various hardware components in the final stages.



Fig.4.21 *Soldering Kit*

* **Laptop or a host machine:** From the beginning of the project to the demonstration with a plug and play model, a laptop with all the required softwares installed and with a stable internet connection is a must. Writing the code, producing a file for the microcontroller, simulation and design of the project and much more is impossible without a laptop.

**Software Tools**

* **EasyEDA for schematic and PCB design :** It is a linux operating systems based open source schematic and PCB designer. It allows users to model and document their prototypes. It enables the analysis and experimentation of the design before making a permanent circuit. The software facilitates a project to be viewed and exported in three designs that is breadboard, PCB and schematic.
* **Eclipse IDE :** It is one of the most popular Integrated Development Environment used for programming in languages like C, C++, Java etc. In this project it plays an integral part in the development of software. Its workspace enables users to handle all the projects at the same place. The testing and debugging, for the identification of any bugs is relatively easy in this IDE.
* **Compiler:** The source code is written in a high level programming language. The compiler converts the source code written into a machine level language.
* **Debugger:** It is a computer program which helps in testing and finding bugs(errors) in a software code. It plays an important part in pointing errors and its effective rectification.
* **Assembler:** It converts the assembly language into a binary machine code which can be processed by the CPU.
* **Linker:** It is a computer program that links one or more object files and combines them into a single executable file, library file or object file.
* **Geany Text editor:** It is a lightweight GUI text editor based on Scintilla and GTK. It is now ported to almost all operating systems like Linux, Windows, Mac, Solaris etc. The advantage of this editor is its GUI, short loading time,support cross platform and flexibility.
* **Arduino IDE:** It is an open source cross platform IDE that facilitates to write code and upload it to the board. It can be used with any arduino boards making that its biggest advantage. It can be used with windows, Mac and Linux operating systems. In this project it is used for the programming of the arduino board.
* **Teraterm:** It is an open source and free application which emulates computer terminals. In other words it is a serial terminal monitor. It supports telnet, SSH and ensures serial port connections. It has a macro scripting language and ensures high terminal standards.

**Cloud Service - Thingspeak**

ThingSpeak is an open-source Internet of things (IoT) application and API. It uses the HTTP and MQTT protocol to store and retrieve data over the internet or via a Local Area Network (LAN). It can generate instant live data visualizations and send notifications using web services. ThingSpeak allows users to aggregate, visualize, and analyze live data streams in the cloud. It has integrated support of MATLAB from MathWorks. User can send:

* Data to ThingSpeak from their devices
* Create instant visualization of live data
* Send alerts.

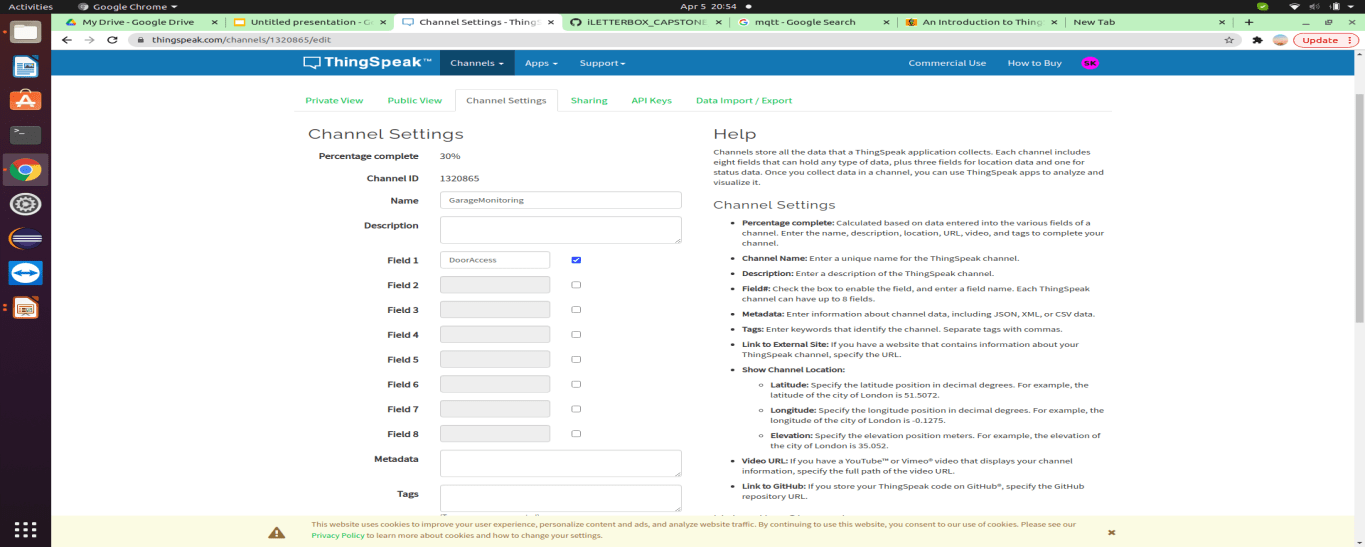
****

Fig.4.22 *Thingspeak channel created for the project*

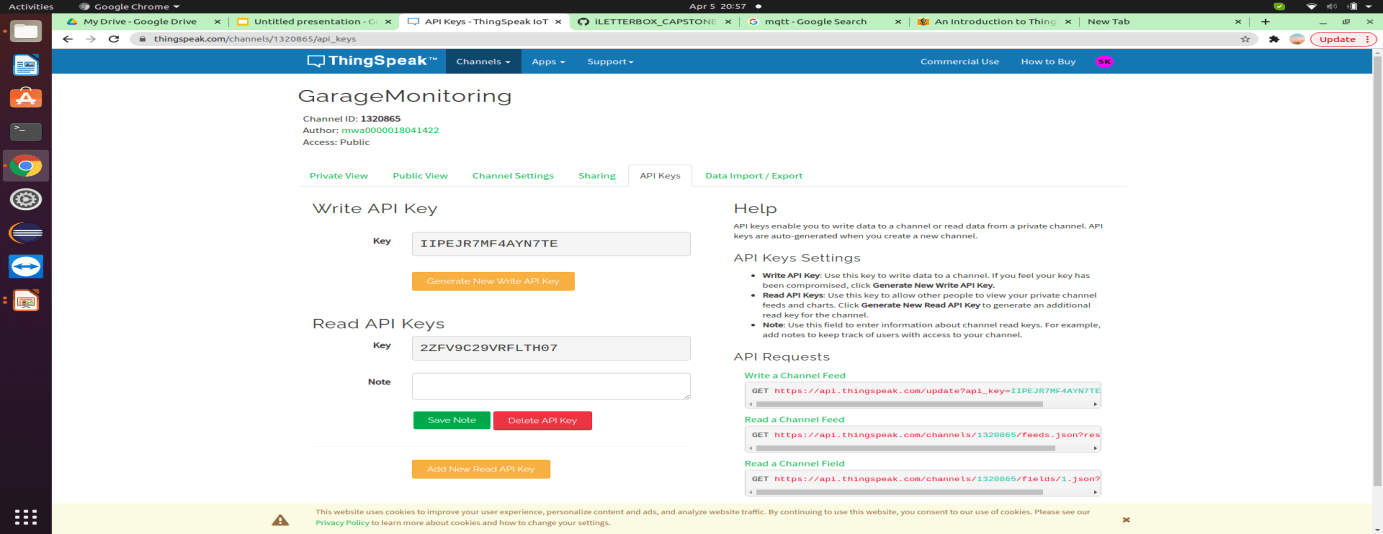
****

Fig.4.23 *Setting Channel for Read and Write API*

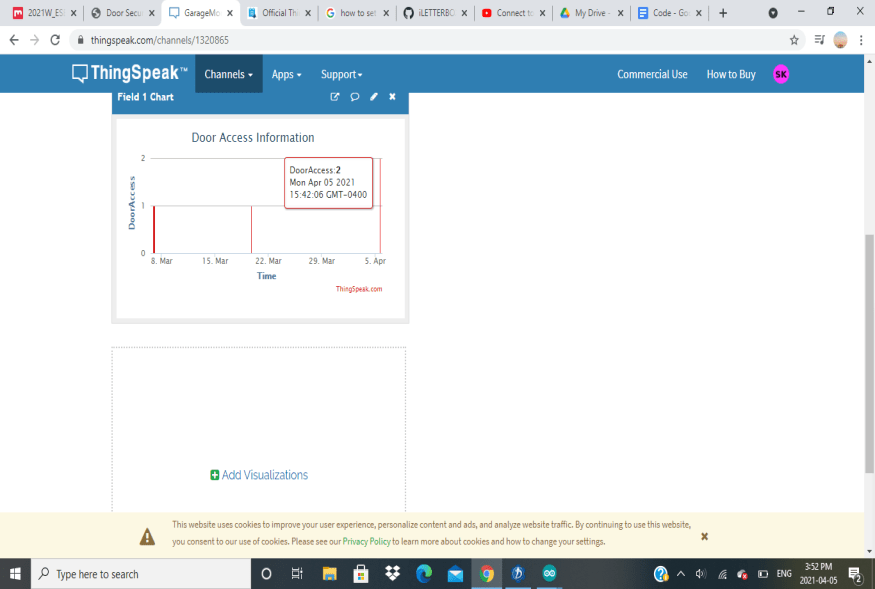
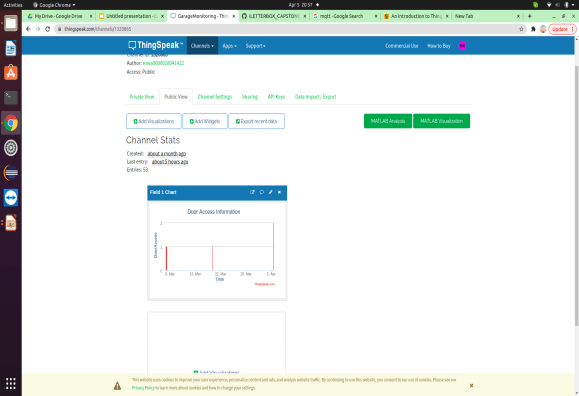


Fig.4.24 *Visualization in Thingspeak*

**PCB Design**

PCB design for our project is done on EasyEDA. It is an easier and powerful online PCB design tool that allows to design and share projects. Printable PCB layer image output is also supported in PDF, PNG, and SVG formats. Circuit designing is completed by using manual routing. We used the same schematics which we used to understand how we can draw out the connection between the Beaglebone Black and Sensors/ Devices.

Easy EDA has a large user contributed library thus people can easily find almost any component which they require for the project. To get a successful PCB design from a given schematic one just needs to be sure that the component has a suitable footprint else the design wouldn’t be possible. One more advantage is, it can support other software libraries and schematics which includes Altium, EAGLE, LTspice, and DXF.

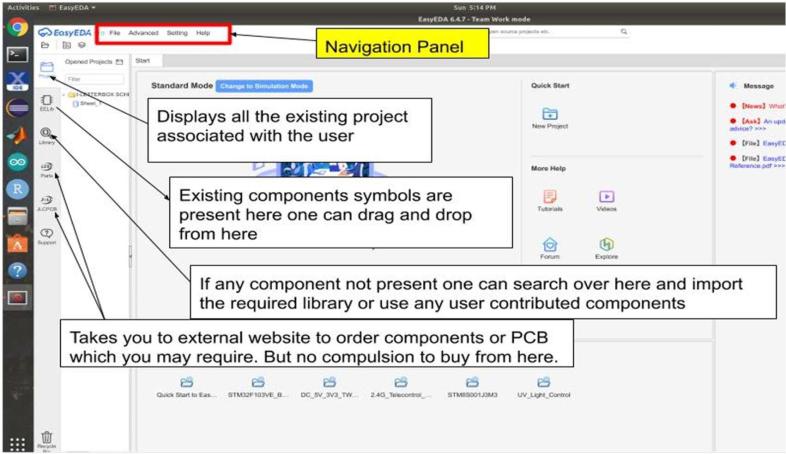


Fig.4.25 *EasyEDA* *Introduction window*

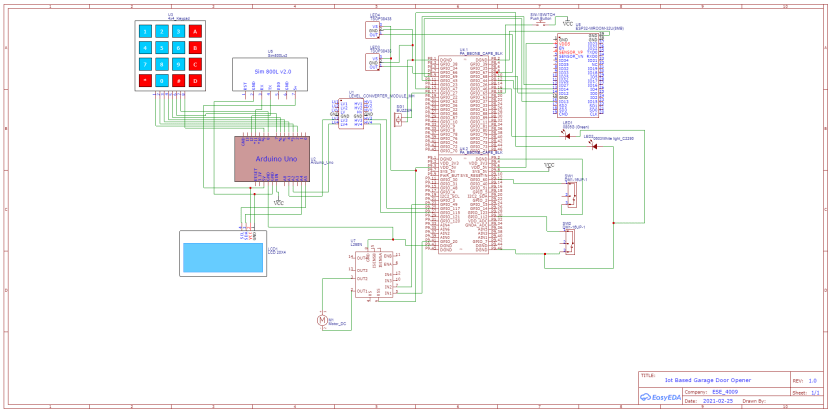


Fig.4.26 *Circuit Schematic Drawing*

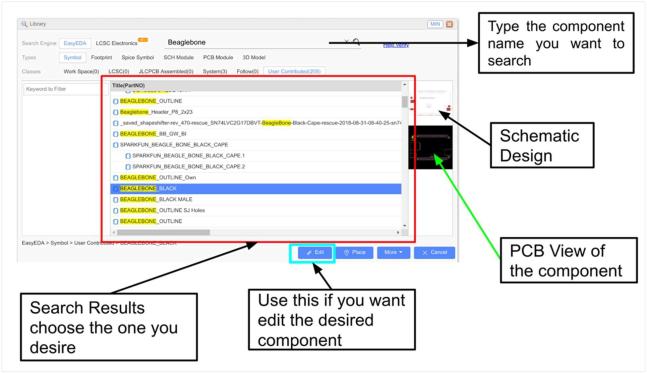


Fig.4.27 *Search Window for Components*

Upon following the user manual guidelines for this software, we finally drew the PCB design for our project as shown below. Through this software one can also generate the BOM (Bill of Material) for the project which will give you a list of components to be purchased to make this project. This BOM can be used as a checklist when one goes to the market to get the components.

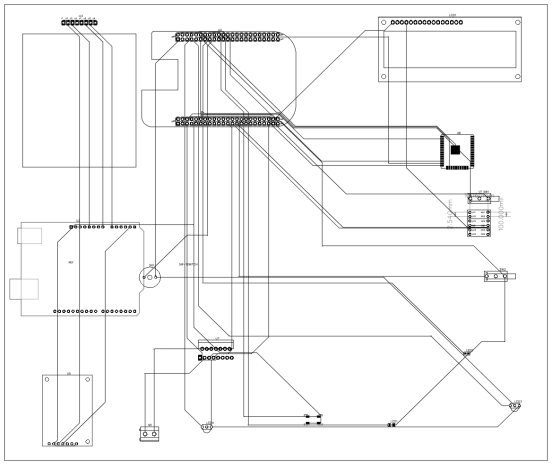


Fig.4.28 *PCB Design for the project*

**Chapter V**

**Implementation and Test**

**Flow Chart**

**1.Flow Chart for IoT based smart garage door opening and closing using Webpage:**

**Explanation:**

First the local server is started and the IP address is entered to open the login page for the user. If the correct login id and password details are given the provision will be provided to open or close the garage door. If the user enters the incorrect details the login page access will be denied. Once the correct login details are entered a choice will be given for opening or closing the door. If the user chooses the open icon, an input will be given for the relay to operate which will in turn enable the rotation of the dc motor and the garage door will be opened. After the door is opened a notification is sent to the user with the help of a GSM module and the door opening time gets recorded in the cloud. If the user forgets to close the door for a set period of time, a notification will be sent to the user to remind them that the door is not closed. The next stage of this project is if the choice is made by the user for closing the garage door. In our project for ensuring a secure and safe closing of the door an IR proximity sensor along with limits switch is used. In case an obstacle for example: a kid comes in between when the door is being closed the two components prevent the door from closing. If an obstacle is detected, a buzzer will be activated which will notify the user of the potential occurrence of a danger. Once the obstacle is cleared by the user, the closing of the door will be possible by pressing the key of the keypad. If an obstacle is not detected it will close normally as designed. Once the door is closed, a message will be sent to the user to inform the user about the closing of the door and a record of the door closing time will also be stored in the cloud.

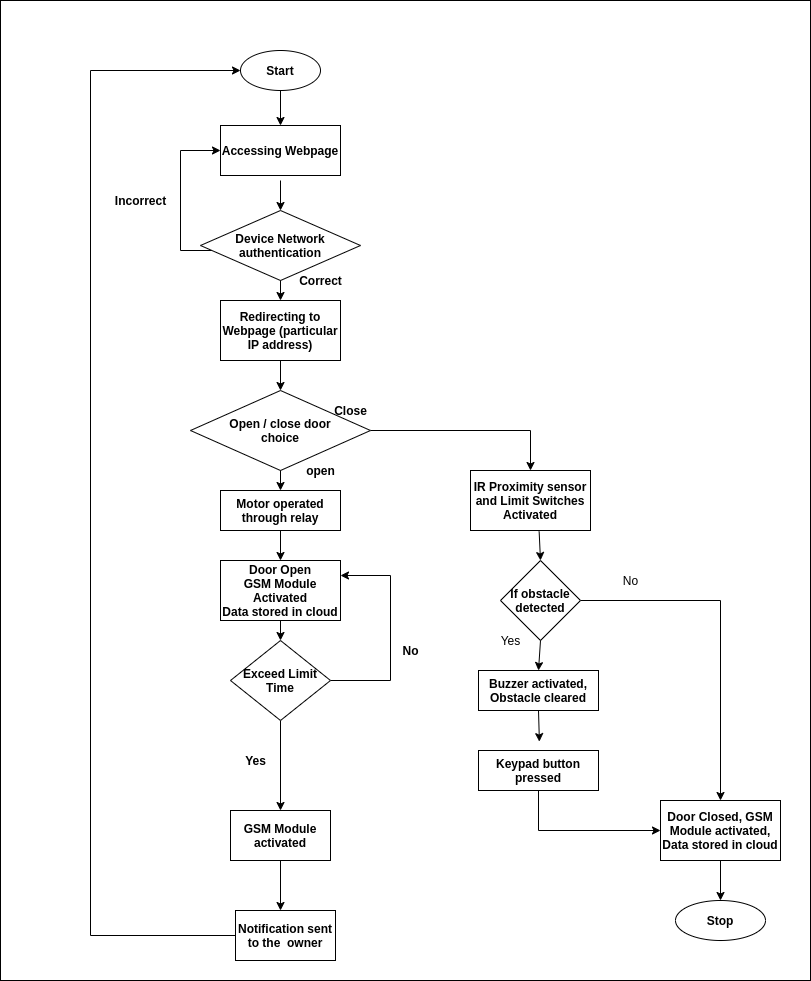
****

Fig5.1. *Flowchart using Webpage*

**2.Flow Chart for IoT based smart garage door opening and closing using keypad and LCD screen:**

**Explanation:**

At the initial stage of the flow chart, when the vehicle will come to enter the garage, the keypad on the door will ask the password to open the door. If the password entered by the user is correct then the motor drivers will start operating through the relay module and the door will open but if the password will be incorrect then the LCD display will show the “ACCESS DENIED”. If a wrong password is entered more than three times via Keypad, A message will be sent to the user via GSM module to notify incorrect password entry, thus it enhances the security feature of the project. After this stage, all the information regarding the vehicle user and the timing will be saved in the cloud for future use or for security purposes as well. If the door will open for so long then the GSM module will activate and it will send the notification to the user through SMS that the door is open, if not then a key is used to close the door. After that, the IR proximity sensor and the limit switch will start functioning and they will detect the obstacle at the time of closing the door. If they sense any obstacle, then the buzzer will start beeping and the door will remain open and then a button will be pressed to close the door again or the closing will be pressed. If there will be no object or obstacle then the door will close and all the data regarding the entry and exit of the object will be saved in the cloud storage. GSM module will enable at the closing of the door as well, if there will be any suspicious activity that will happen to inform the user.

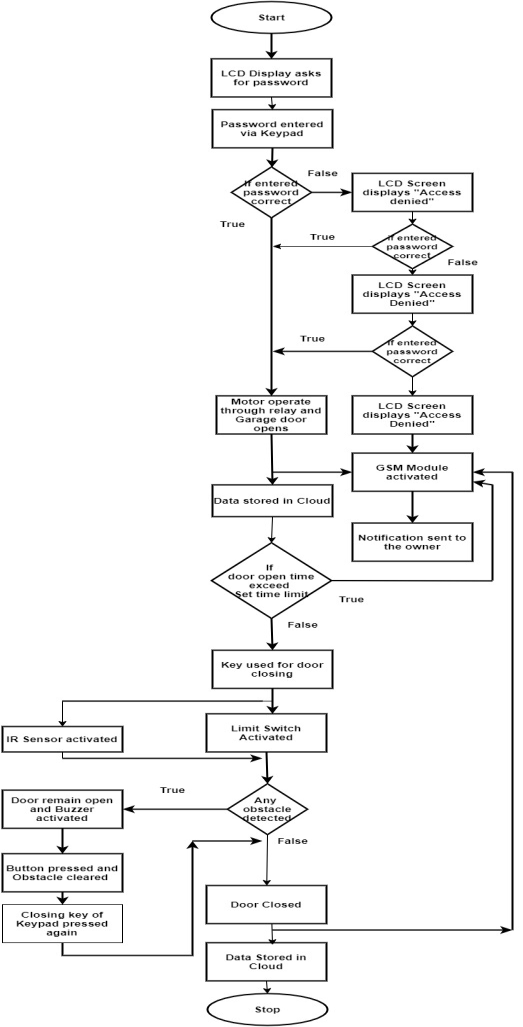


Fig.5.2 *Flowchart using LCD and Keypad*

**Interfacing with BBB**

* **Interfacing of Beaglebone black with IR Proximity sensors**

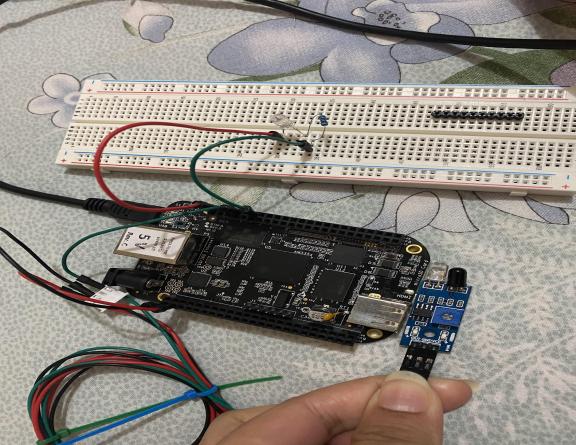
****

Fig.5.3 *Interfacing of BBB with IR Proximity sensors*

Above figure shows how IR proximity sensor is interfaced with Beaglebone black and also one LED is connected only for the demonstration purpose. We have connected the SIG/OUT pin of IR sensor with the GPIO\_66 pin of P8 header of Beaglebone Black and also VCC pin to 3V and Ground pin connected to the Ground pin of BBB.

Code:

|  |
| --- |
| #**include** <stdio.h> //includes standard I/O header files  #**include** <iobb.h> //includes GPIO I/O header files  #**include** <unistd.h> //includes POSIX API library for adding delay in operation  #**define** IRoccupancy 8,8 // PIR sensor connected to pin 8 of header P8  #**define** LEDpin 8,12 //LED is connected to pin 12 of header P8  **int** **main**(**void**) // main function  {  iolib\_init(); // initializing GPIO I/O library  iolib\_setdir(IRoccupancy, DigitalIn); //setting GPIO pin for output  iolib\_setdir(LEDpin, DigitalOut);  **while**(1) //continuous loop  {  **if** (is\_high(IRoccupancy)) //if pin 8 of header is low  {  printf("Garage is Unoccupied \n"); //print on terminal  pin\_low(LEDpin); //LED will not glow  usleep(1200000); //delay operation  }  **if** (is\_low(IRoccupancy)) // if pin 8 of header is high  {  printf("Garage is Occupied \n");  pin\_high(LEDpin); //LED will glow  usleep(1200000);  }  }  iolib\_free(); //end of program (freed library resources)  **return**(0); //return a value  } |

* **Interfacing of ESP32 Wifi Module with BBB**

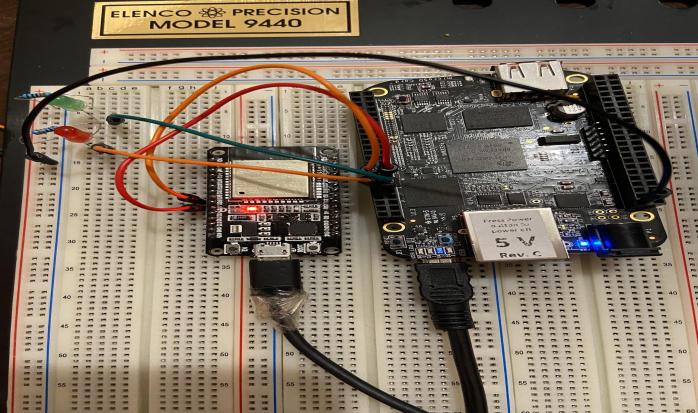
****

Fig.5.4 *Interfacing of ESP32 wifi module with BBB*

Above figure depicts the interfacing of BBB with ESP32 wifi module and also we have used two LEDs for the demonstration purpose. The connections between both the components are explained below:

**Connections of Beaglebone Black and LEDs**

* Green LED is connected to P8.11 of Beaglebone Black.
* Red LED is connected to P8.12 of Beaglebone Black.
* Resistors of 330k Ohms have been used in between both the LEDs.
* GND has also been connected.

Code:

|  |
| --- |
| #**include** <iobb.h>  #**include** <stdio.h>  #**define** GreenLED 8,16  #**define** RedLED 8,15  #**define** Gon 8,11  #**define** Ron 8,12  **int** **main**(**void**)  {  iolib\_init();  iolib\_setdir(GreenLED,DigitalIn);  iolib\_setdir(RedLED,DigitalIn);  iolib\_setdir(Gon,DigitalOut);  iolib\_setdir(Ron,DigitalOut); //iolib\_delay\_ms(500);  **while**(1)  {  **if** (is\_high(GreenLED))  {  printf("Green LED is ON \n");  pin\_high(Gon); iolib\_delay\_ms(500);  }  **else** (is\_low(GreenLED))  {  printf("Green LED is OFF\n");  pin\_low(Gon); iolib\_delay\_ms(500);  }  **if** (is\_high(RedLED))  {  printf("Red LED is ON \n");  pin\_high(Ron); iolib\_delay\_ms(500);  }  **else** (is\_low(RedLED))  {  printf("Red LED is OFF \n");  pin\_low(Ron); iolib\_delay\_ms(500);  }  }  iolib\_free();  **return**(0);  } |

* **Interfacing BBB with Motor Driver L298N and connecting it to DC motor**

**A picture containing electronics, cable, connector, adapter

Description automatically generated**

Fig.5.5 *Interfacing BBB with Motor Driver L298N and connecting it to DC motor*

Above figure shows how BBB is connected to the L298N motor driver and connecting to the DC motor. We have used Pin no. 23 and 41 of P9 header for the interfacing. The connections are explained below:

Motor Driver 12V to Battery +

Motor Driver Ground to Battery -

Motor Driver Ground to Beaglebone DGND

Motor Driver Input 1 to Beaglebone P9.41

Motor Driver Input 2 to Beaglebone P9.23

Motor Wire 1 to Motor A+

Motor Wire 2 to B+

Code:

|  |
| --- |
| #**include** <stdio.h> // Standard Input Output Library  #**include** <iobb.h> //GPIO Library  **int** motor1 = 23; //Motor 1 Pin  **int** motor2 = 41; //Motor 1 Pin  **int** port9 = 9;//Defining variable for port 9  **int** port8 = 8;//Defining variable for port 8  **char** motion; //Variable to take input  **int** **main**(**void**)  {  iolib\_init(); //initialize the iobb library  iolib\_setdir(port9, motor1, DigitalOut);//setting the pin as Digital out  iolib\_setdir(port9, motor2, DigitalOut);//Setting the pin as Digital out  **while**(1)  {  printf("Enter 0: Stop, 1: CW, 2: CCW: ");//print message to the user  motion = getchar();//Read one character input  **if**(motion == '1') //if input is 1  {  printf("Motor in CW direction\n"); //print message to the user  pin\_high(port9, motor1); //INA pin high  pin\_low(port9, motor2); //INB pin low  }  **else** **if**(motion == '2') //if input is 2  { printf("Motor in CCW direction\n");//print message to user  pin\_low(port9, motor1); //INA pin low  pin\_high(port9, motor2); //INA pin high  }  **else** **if**(motion == '0') //if input is 0  {  printf("Motor Stopped\n"); //print message to user  pin\_low(port9, motor1); //INA pin low  pin\_low(port9, motor2); //INB pin low  }  }  iolib\_free(); // Release all GPIOs to default status  **return**(0);  } |

* **Interfacing of Limit Switch with BBB**

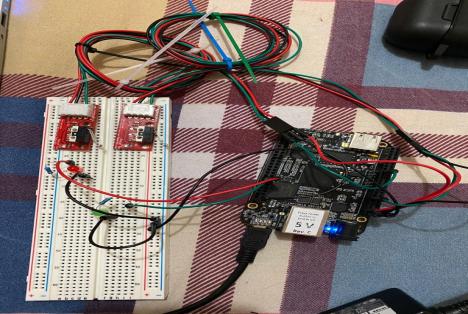
****

Fig.5.6 *Interfacing of Limit Switch with BBB*

The above figure illustrates the connection between the BBB and limit switches. In our project, we are using limit switches to detect the status of the garage door and if that door opens too long and we can use limit switches manually to close the door.

* Connected green led to the pin 16 of header P8
* Connected red led to the pin 15 of header P8
* Connected limit switch for door open to the pin 30 of header P9
* Connected limit switch for door close to the pin 12 of header P9

Code:

|  |
| --- |
| #**include** <iobb.h> //includes GPIO I/O header files  #**include** <stdio.h> //includes standard I/O header files  #**include** <unistd.h> //includes POSIX API library for adding delay in operations  #**define** GreenLED 8,16 //Green led connected to pin 16 of header P8  #**define** RedLED 8,15 //Red led connected to pin 15 of header P8  #**define** LMcheckDoorOpened 9,30 //Limit switch for door open connected to pin 30 of header 9  #**define** LMcheckDoorClosed 9,12 //Limit switch for door close connected to pin 12 of header 9  **int** **main**(**void**) // main function  {  iolib\_init(); // initializing GPIO I/O library  iolib\_setdir(GreenLED, DigitalOut); //setting GPIO pin for Output  iolib\_setdir(RedLED, DigitalOut);  iolib\_setdir(LMcheckDoorOpened, DigitalIn); //setting GPIO pin for Input  iolib\_setdir(LMcheckDoorClosed, DigitalIn);  **while**(1) //continuous loop  {  **if** (is\_low(LMcheckDoorOpened)) //if pin 30 of header is low  {  printf("Door is open \n"); //print on terminal  pin\_high(RedLED); //Red LED will glow  usleep(1200000); //delay operation  }  **else** **if** (is\_high(LMcheckDoorOpened)) //if pin 30 of header is high  {  pin\_low(RedLED); //Red LED will not glow  }  **if** (is\_low(LMcheckDoorClosed)) // if pin 12 of header is low  {  printf("Door is Closed \n");  pin\_high(GreenLED); //LED will glow  usleep(1200000);  }  **else** **if** (is\_high(LMcheckDoorClosed)) //if pin 12 of header is high  {  pin\_low(GreenLED); // LED will not glow  } }  iolib\_free(); //end of program (freed library resources)  **return**(0); //return a value  } |

* **Interfacing Buzzer with BBB**

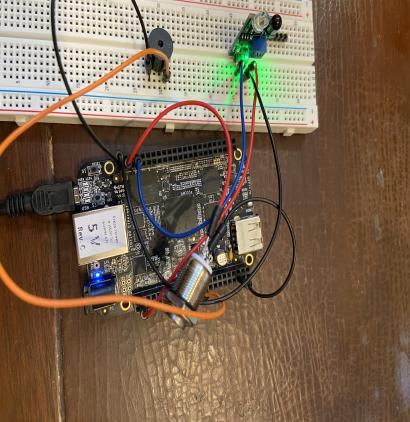
****

Fig.5.7 *Interfacing Buzzer with BBB*

In this interfacing we have used IR sensor and push button as well. The purpose of using IR sensor is to detect any obstacle and push button is to disable the buzzer manually. The connection between all the components explained below:

**Connections of Buzzer with Beaglebone Black:**

* The positive (+ve) pin of Buzzer is connected with pin 23 of header 9 of beaglebone black.
* The negative (-ve) pin of Buzzer has been grounded.

**Connections of IR Sensor with Beaglebone Black:**

* The output pin of IR Sensor is connected to pin 7 of header 8 of beaglebone black.
* Ground pin of both the components has been connected to each other.
* 5V power supply has been provided to the Vcc pin of IR Sensor.

**Connections of Push Button with Beaglebone Black:**

* One pin of the push button is connected to pin 41 of header 9 of beaglebone black.
* 3.3V power supply has been provided to the other pin of the push button.

Code:

|  |
| --- |
| #**include** <stdio.h> //includes standard I/O header files  #**include** <iobb.h> //includes GPIO I/O header files  #**include** <unistd.h> //includes POSIX API library for adding delay in operation  #**define** Buzzer 9,23 //pin 23 of header 9  #**define** PushButton 9,41 //pin 41 of header 9  #**define** IRbtwnDoor 8,7 //pin 7 of header 8  **int** **main**(**void**) // main function  {  iolib\_init(); // initializing GPIO I/O library  iolib\_setdir(IRbtwnDoor, DigitalIn); //setting GPIO pin of IR sensor for input  iolib\_setdir(Buzzer, DigitalOut); //setting GPIO pin of Buzzer  iolib\_setdir(PushButton, DigitalIn); //setting GPIO pin of push button for input  **while**(1) //continuous loop  {  **if** (is\_high (IRbtwnDoor)) //if pin 7 of header 8 is low  {  printf("Door Closed \n"); //print on terminal  pin\_low(Buzzer); // Buzzer remains disabled  usleep (1200000); //delay operation  }  **if** (is\_low (IRbtwnDoor)) //if pin 7 of header 8 is high  {  printf("Obstacle \n"); //print on terminal  pin\_high(Buzzer); // Buzzer will be enabled  usleep (1200000); //delay operation  **if** (is\_high (PushButton)) //when button is pressed  {  printf("Buzzer is off \n"); //print on terminal  pin\_low(Buzzer); //Buzzer will be disabled  usleep (1200000); //delay operation  } } }  iolib\_free(); //end of program (free library resources)  **return**(0); //return a value  } |

* **Interfacing Button with BBB**

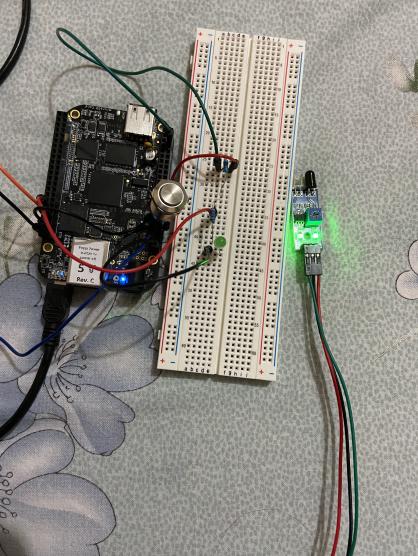
****

Fig.5.8 *Interfacing of Button with BBB*

Above figure shows the interfacing of the push button with BBB along with the IR sensor and LED. The hardware connections of this interfacing are explained below:

**Push Button with Beaglebone Black:**

* One pin of the push button has been connected to pin 41 of port 9 (P9.41) of Beaglebone Black.
* The resistor of 10K has been connected in between push button and Beaglebone Black.
* The 3.3V power supply has been connected to the other pin of the push button.

**Connections of Beaglebone Black with LED:**

LED has been connected to the beaglebone black for the demonstration.

* The positive(+) pin of LED has been connected to pin 12 of port 8(P8.12) of beaglebone black.
* The negative (-) pin of the LED has been grounded.

**Connection of IR sensor and Beaglebone Black:**

* The output pin of IR Sensor is connected to pin 7 of header 8 (P8.7) of beaglebone black.
* The 5V power supply is connected to the Vcc pin of the IR sensor.
* Both GND pins of the IR sensor and beaglebone black are connected to each other.

Code:

|  |
| --- |
| #**include** <stdio.h> //includes standard I/O header files  #**include** <iobb.h> //includes GPIO I/O header files  #**include** <unistd.h> //includes POSIX API library for adding delay in operation  #**define** GreenLED 8,12 //pin 12 of header 8  #**define** PushButton 9,41 //pin 41 of header 9  #**define** IRbtwnDoor 8,7 //pin 7 of header 8  **int** **main**(**void**) // main function  {  iolib\_init(); // initializing GPIO I/O library  iolib\_setdir(IRbtwnDoor, DigitalIn); //setting GPIO pin of IR sensor for input  iolib\_setdir(GreenLED, DigitalOut); //setting GPIO pin of LED  iolib\_setdir(PushButton, DigitalIn); //setting GPIO pin of push button for input  **while**(1) //continuous loop  {  **if** (is\_high (IRbtwnDoor)) //if pin 7 of header 8 is low  {  printf("Door Closed \n"); //print on terminal  pin\_low(GreenLED); // LED will not glow  usleep (1200000); //delay operation  }  **if** (is\_low (IRbtwnDoor)) //if pin 7 of header 8 is high  {  printf("Obstacle \n"); //print on terminal  pin\_high(GreenLED); // LED will glow  usleep (1200000); //delay operation  **if** (is\_high (PushButton)) //when button is pressed  {  printf("Buzzer is off \n"); //print on terminal  pin\_low(GreenLED); // LED will not glow  usleep (1200000); //delay operation  } } }  iolib\_free(); //end of program (free library resources)  **return**(0); //return a value  } |

* **Interfacing Arduino with BBB**

****

Fig.5.9 *Interfacing of Arduino with BBB*

Above figure explains the hardware connections between Arduino and Beaglebone black. In this interfacing, we have connected both the controllers through UART communication protocol. The connections between both the components are explained below:

* Pin 2 of Arduino is connected to the UART4\_RXD pin of Beaglebone Black.
* Pin 3 of Arduino is connected to the UART4\_TXD pin of Beaglebone Black.
* Ground pin is connected to the ground pin of Beaglebone Black.

Code:

|  |
| --- |
| #**include** <stdio.h> //Standard C input Output Library  #**include** <unistd.h> //To use functions like File Opening, Closing, sleep etc.  #**include** <termios.h> // For Seeting up UART communication  #**include** <fcntl.h> // File control functions like O\_RDWR, O\_NONBLOCK etc.  #**include** <string.h> //For String Operations  **int** **main** ()  {  **char** inArray[100]; // declare a char array for receiving data  **char** outArray[100]; // A char array to store and send data  **int** file = open("/dev/ttyO4", O\_RDWR | O\_NONBLOCK);// Open file in Read Write & Non Blocking Mode  **if** (file > 0) { printf("Started UART communication\n");  }  **else** { printf("Some Error Occured\n"); **return** 0;// Stop the code  }  **struct** **termios** **uartsetting**;  uartsetting.c\_cflag = B9600 | CS8 | CREAD; //Set Baudrate, Byte size and Reading enabled uartsetting.c\_lflag = ICANON; //Canonical Mode- Input is available line by line  tcsetattr(file,TCSANOW,&uartsetting); //Apply the setting to UART  tcflush(file, TCIOFLUSH); //Flush all Input and Output Data  **while**(1) // An Infinite Loop  {  outArray[0] = '\0';  strcpy(outArray, "Send Some Message\n"); // Copy a string in char array  write(file, outArray, strlen(outArray)); // Sending message  printf("Data Sent \n");  sleep(1);  **int** inputNum = read(file,&inArray,100); // Read the input Message  **if**(inputNum > 0)  {  **for**(**int** i=0; i < inputNum; i++)  { printf("%c",inArray[i]); //print input message byte by byte }  printf("\n");  sleep(1);  } }  close(file); //Close the f1 at last  } |

* **Interfacing LCD display with Arduino**

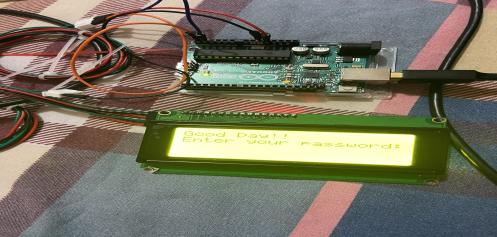
****

Fig.5.10 *Interfacing LCD display with Arduino*

This figure shows the interfacing of LCD display with an Arduino. In our project we are using a LCD Module to display a message to the user while the user wants to open the door manually. The LCD module will display a message, for instance, Enter Your Password. The user will enter the password using the keypad and thus the door will open. However, if the user enters the wrong password the LCD will display another message. This interface will allow the user to open the door manually in case the user does not have the access to their mobile phone. This interfacing makes the project more diverse. The hardware connections between both the components are explained below:

* Vcc pin of lcd is connected to 5v of arduino
* Ground pin of Lcd is connected to ground pin of arduino
* SDA pin of lcd is connected to A4 and SCL to A5

Code:

|  |
| --- |
| #**include** <LiquidCrystal\_I2C.h> // Including libraries for the lcd\_i2c  #**include** <LCD03.h> // Including libraries for lcd module  #**include** "Wire.h" // including libraries for I2C module  //Set the pins on the I2C chip used for LCD connections //ADDR,EN,R/W,RS,D4,D5,D6,D7  LiquidCrystal\_I2C **lcd**(0x27,2,1,0,4,5,6,7);  **void** **setup**() // function created to run the code as soon as the arduino is powered up  {  lcd.begin (20,4); // Defining pins of 20 x 4 LCD module  lcd.setBacklightPin(3,POSITIVE); // To set the backlight of the lcd as high  lcd.setBacklight(HIGH); // to set the backlight  lcd.print("Good Day!!"); // this command will print the mentioned text on LCD screen  lcd.setCursor(0,1); // to move to the next line; beacuse lcd has 20 rows and 4 column  lcd.print("Enter your password:"); // this command will print the mentioned text on LCD screen  }  **void** **loop** () // the loop function does precisely what its name suggests and loops consecutively allowing your program to change and respond. We use to actively control the arduino board.  { } |

* **Interfacing of Arduino with Keypad**

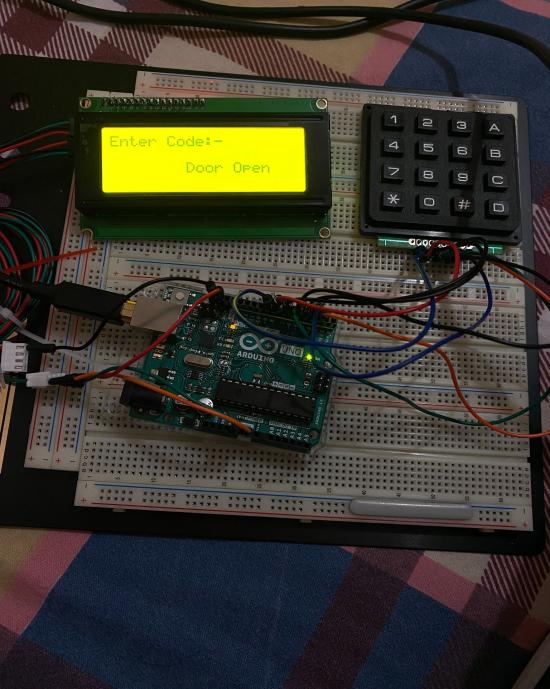
****

Fig.5.11 *Interfacing of Arduino with Keypad*

In this interfacing, we have used LCD display as well to show the status of the garage door or showing the password. We have used a keypad to enter the password for opening the door. It has 8 terminals, out of 8 four are rows and other four are columns, which we have connected with the Arduino pins.

Code:

|  |
| --- |
| #**include** "Wire.h" // For I2C communication  #**include** "LCD.h" // For LCD  #**include** "LiquidCrystal\_I2C.h" // For I2C LCD display  #**include** "Adafruit\_Keypad.h" // keypad library for arduino  #**define** KEYPAD\_PID3844 // define the 4x4 matrix keypad  #**define** R1 5 //connected pin 5 of arduino with pin 1 of ROW1  #**define** R2 4 //connected pin 4 of arduino with pin 2 of ROW2  #**define** R3 3 //connected pin 3 of arduino with pin 3 of ROW3  #**define** R4 2 //connected pin 2 of arduino with pin 4 of ROW4  #**define** C1 9 //connected pin 9 of arduino with pin 1 of COLUMN1  #**define** C2 8 //connected pin 8 of arduino with pin 2 of COLUMN2  #**define** C3 7 //connected pin 7 of arduino with pin 3 of COLUMN3  #**define** C4 6 //connected pin 6 of arduino with pin 4 of COLUMN4  // leave this import after the above configuration  #**include** "keypad\_config.h" // for keypad configuration  Adafruit\_Keypad customKeypad = Adafruit\_Keypad( makeKeymap(keys), rowPins, colPins, ROWS, COLS);  LiquidCrystal\_I2C **lcd**(0x27, 2, 1, 0, 4, 5, 6, 7); // 0x27 is the default I2C bus address of the backpack  **int** max\_attempts =3; // define the no. of password attempts  **int** attempts =0; // initially 0  **void** **setup**()  {  Serial.begin(9600); // serial baud rate  customKeypad.begin(); // Set off LCD module  lcd.begin (20, 4); // 20x4 LCD module  lcd.setBacklightPin(3, POSITIVE); // BL, BL\_POL  lcd.setBacklight(HIGH); //to set the backlight  lcd.setCursor(0, 0); // setting the writing position  lcd.print(" Welcome to Garage"); // print on the LCD screen  lcd.setCursor(0, 1); // setting the writing position  lcd.print(" Monitoring System."); // print on the LCD  lcd.setCursor(0, 2); // setting the writing position  delay(5000); // delay operation  lcd.clear(); // clear the screen  lcd.setCursor(0,0); // setting the writing position with row and column position  lcd.print("Enter Code:-"); // Enter the password  lcd.setCursor(0, 1); // to move the cursor  lcd.print(" "); // print on LCD  }  **char** GetKey; //declaration for characters  **char** KEY[4] = {'1', '2', '3', '4'}; // Array declaration  **char** Check[4]; **int** c = 0; // set initial value 0  **void** **loop**() // loop begins  **if** (c == 1) // if integer c is 1  {  lcd.setCursor(0, 1); // first row of lcd  lcd.print(GetKey); // print on the LCD  Check[0] = GetKey; //entered value assigns to the array address 0  }  **else** **if** (c == 2)  {  lcd.setCursor(1, 1); // first row second column of lcd  lcd.print(GetKey); // print on LCD  Check[1] = GetKey; // value assign to the array address 1  }  **else** **if** (c == 3)  {  lcd.setCursor(2, 1); // first row third column of LCD  lcd.print(GetKey); // print on the LCD  Check[2] = GetKey; // value assign to the array address 2  }  **else** **if** (c == 4)  {  lcd.setCursor(3, 1); // first row fourth column of LCD  lcd.print(GetKey); // print on the LCD  Check[3] = GetKey; // value assign to the array address 3  c = 0;  **if** (Check[0] == KEY[0] && Check[1] == KEY[1] && Check[2] == KEY[2] && Check[3] == KEY[3]) // if entered values are same  {  lcd.setCursor(0, 1);  lcd.print(" ");  lcd.setCursor(0, 2); // second row of LCD  lcd.print("Door Open"); // print Door Open on the LCD  lcd.setCursor(0, 1);  lcd.print(" ");  }  **else**  {  attempts=attempts+1; // if attempts exceeds more than 3  **if** (attempts >= max\_attempts)  {  lcd.clear(); // clear the LCD  lcd.setCursor(0,2);  lcd.print("Alert send"); // alert notification send  delay(5000); // delay operation  attempts=0;  }  lcd.setCursor(0, 2);  lcd.print("Wrong Password"); // print wrong password  lcd.setCursor(0, 3);  lcd.print("Max 3 attempts"); //print maximum 3 attempts on LCD  delay(5000); // delay operation  lcd.clear();  lcd.setCursor(0,0);  lcd.print("Enter Code:-"); //again ask for password  lcd.setCursor(0, 1);  lcd.print(" ");  } }  customKeypad.tick();  **while** (customKeypad.available()) // continuous loop begins  {  keypadEvent e = customKeypad.read(); //read the keypad entry  Serial.print((**char**)e.bit.KEY);  **if** (e.bit.EVENT == KEY\_JUST\_PRESSED) // if condition starts  { Serial.println(" pressed"); // shows pressed status  GetKey = (**char**)e.bit.KEY; // character key  **if** (GetKey == 'D') // if key is D (alpha numeric key)  { lcd.setCursor(0, 3);  lcd.print(" ");  lcd.setCursor(0, 3);  lcd.print("DOOR CLOSING"); // print Door closing on the LCD  delay(2000); //delay operation  lcd.setCursor(0, 3);  lcd.print(" "); }  **else** c++; }  **else** **if** (e.bit.EVENT == KEY\_JUST\_RELEASED)  Serial.println(" released"); // key released } }  Keypad\_config.h // This file contains predefined setup for various Adafruit Matrix Keypads.  #**ifndef** \_\_KEYPAD\_CONFIG\_H\_\_  #**define** \_\_KEYPAD\_CONFIG\_H\_\_  #**if** defined(KEYPAD\_PID3844)  **const** byte ROWS = 4; // four rows  **const** byte COLS = 4; // four columns // define the symbols on the buttons of the keypads  **char** keys[ROWS][COLS] = {{'1', '2', '3', 'A'}, {'4', '5', '6', 'B'}, {'7', '8', '9', 'C'}, {'\*', '0', '#', 'D'}};  byte rowPins[ROWS] = {R1, R2, R3, R4}; // connect to the row pinouts of the keypad  byte colPins[COLS] = {C1, C2, C3, C4}; // connect to the column pinouts of the keypad  #**endif**  #**if** defined(KEYPAD\_PID1824) || defined(KEYPAD\_PID3845) || defined(KEYPAD\_PID419)  **const** byte ROWS = 4; // rows  **const** byte COLS = 3; // columns // define the symbols on the buttons of the keypads  **char** keys[ROWS][COLS] = { {'1', '2', '3'}, {'4', '5', '6'}, {'7', '8', '9'}, {'\*', '0', '#'}};  byte rowPins[ROWS] = {R1, R2, R3, R4}; // connect to the row pinouts of the keypad  byte colPins[COLS] = {C1, C2, C3}; // connect to the column pinouts of the // keypad  #**endif**  #**if** defined(KEYPAD\_PID1332)  **const** byte ROWS = 1; // rows  **const** byte COLS = 4; // columns // define the symbols on the buttons of the keypads  **char** keys[ROWS][COLS] = { {'1', '2', '3', '4'}, };  byte rowPins[ROWS] = {R1}; // connect to the row pinouts of the keypad  byte colPins[COLS] = {C1, C2, C3, C4}; // connect to the column pinouts of the keypad  #**endif**  #**endif** |

* **Interfacing Arduino with GSM Module**

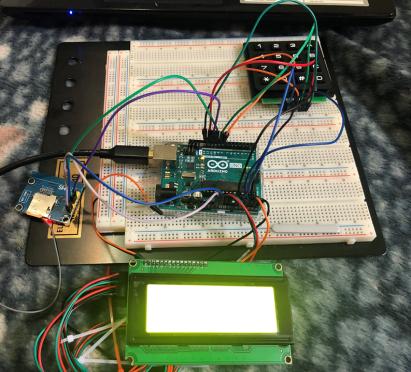
****

Fig.5.12 *Interfacing Arduino with GSM Module*

Above figure shows the interfacing between Arduino and GSM module and also we have connected LCD and Keypad in this interfacing. The purpose of this interfacing is to send an alert message to the owner through the GSM module if an unknown person attempts to access the site by entering the wrong password more than three times.

**Connections between Keypad and Arduino:**

* C1 to C4 pins of keypad connected to A0 to A4 of Arduino
* R1 pin connected to Pin 8 of Arduino
* R2 pin connected to Pin 9 of Arduino
* R3 pin connected to Pin 12 of Arduino
* R4 pin connected to Pin 13 of Arduino

**Connections between LCD and Arduino:**

* Vcc -> Vin of Arduino
* Gnd -> Gnd of Arduino
* Scl -> A5 pin of Arduino
* Sda -> A4 pin of Arduino

**Connections between GSM module and Arduino**

* Vin ----> 5V of Arduino
* Tx -----> Pin 10 of Arduino
* Rx -----> Pin 11 of Arduino
* Gnd ---> Gnd pin of Arduino

Code:

|  |
| --- |
| #**include** <Wire.h>  #**include** <LiquidCrystal\_I2C.h>  #**include** <Keypad.h>  #**include** <SoftwareSerial.h>  SoftwareSerial **mySerial**(10, 11); // RX, TX  **const** byte ROWS = 4; //four rows  **const** byte COLS = 4; //three columns  **char** keys[ROWS][COLS] = { {'1','2','3','A'}, {'4','5','6','B'}, {'7','8','9','C'}, {'\*','0','#','D'} };  byte rowPins[ROWS] = {4,5,6,7}; //connect to the row pinouts of the keypad  byte colPins[COLS] = {8,9,12,13}; //connect to the column pinouts of the keypad  byte rowPins[ROWS] = {8,9,12,13}; //connect to the row pinouts of the keypad  byte colPins[COLS] = {A0,A1,A2,A3}; //connect to the column pinouts of the keypad  Keypad keypad = Keypad( makeKeymap(keys), rowPins, colPins, ROWS, COLS );  LiquidCrystal\_I2C **lcd**(0x27,16,2); // set the LCD address to 0x27 for a 16 chars and 2 line display  String password = "2356"; // set password  String message = "Many Wrong Password Retries";  String number = "+16478075601"; // number set for GSM notifications  **int** sendsms; //Initializing an integer variabe sendsms  String input; //initializing a string variable  **void** **setup**()  {  Serial.begin(9600); //Begin the host machine communication at baud rate 9600  mySerial.begin(9600); //Begin the Arduino communication at Baud rate 9600  Serial.println("STARTING");// Print starting on the serial monitor  lcd.init(); // initialize the lcd // Print a message to the LCD.  lcd.backlight();  lcd.clear(); // clear the LCD screen  lcd.setCursor(0,0); // Set cursor at 0,0  lcd.print("STARTING"); //Print STARTING on LCD  Serial.println("Initializing GSM"); //Print Initializing GSM on serial monitor  delay(5000); // Boot Up time for GSM  Serial.println("STARTED\n\n"); //Print STARTED on the serial monitor  Serial.println("\n\n\n"); // leave three lines  lcd.clear(); // for clearing LCD  lcd.setCursor(0,0);// set the LCD cursor to 0,0 and print ENTER PASSWORD  lcd.print("ENTER PASSWORD"); //Print ENTER PASSWORD on LCD screen  lcd.setCursor(0,1);// Set the LCD cursor to 0,1  lcd.print(input);// Print the input from the user  }  **void** **loop**()  {  **char** key = keypad.getKey(); //Get the input from the pressed key  **if** (key) { Serial.println(key); // If it is a key, print on the LCD  **if**(key != 'C' **and** key != 'D')//If the key is not C or D  { input += key; // Store the pressed key as Input= Input+Key }  **else** **if**(key == 'C')// If the pressed Key is C, Delete the last pressed key  { input = input.substring(0,input.length()-1); }  **else** **if**(key == 'D') //If the pressed key is D, Check whether it matches with the password  { **if**(input == password) //Checking whether Input is same as set password  { Serial.println("Right Password");// If yes print Right Password on serial monitor  lcd.setCursor(0,1); // Set LCD cursor to 0,1  lcd.print("CORRECT PASSWORD . ");// If yes print Correct password on LCD screen  sendsms = 0; // For correct password sendsms=0 }  **else** { Serial.println("Wrong Password"); //If input is not equal to set password, Print wrong password on the serial monitor  lcd.setCursor(0,1);//set LCD screen to 0,1  lcd.print("WRONG PASSWORD X ");// Print WRONG PASSWORD on the LCD screen as well  Sendsms++; //In this case sendsms= sendsms+1 }  delay(2000);//Give delay  input = "";   **if**(sendsms >= 3)//if sendsms >=3 the GSM module will be activated  { lcd.setCursor(0,1);//set the LCD cursor to 0,1  lcd.print("X SENDING SMS X ");//print SENDING SMS on the LCD screen  SMS();//Send SMS  sendsms = 0; // once an sms is sent set the sendsms to 0 } }  Serial.print("Input: "); //print Input on the serial monitor  Serial.println(input);// receive that input and print it on the monitor  lcd.setCursor(0,1);// setthe LCD cursor to 0,1  lcd.print(input); //print the user input on the LCD screen  lcd.print(" ");//then print space }  delay(20);// give delay }  **int** **SMS**() //function for sending sms  { Serial.print("Sending SMS to : "); //print it on serial monitor  Serial.println(number); //print the number on serial monitor  mySerial.println("AT+CMGF=1"); //Sets the GSM Module in Text Mode  delay(200);  //Serial.println ("Set SMS Number");  mySerial.println("AT+CMGS=\"" + number + "\"\r"); //Mobile phone number to send message  delay(200);  String SMS = "Many Wrong Password Retries";  mySerial.println(SMS);  delay(100);  mySerial.println((**char**)26);// ASCII code of CTRL+Z  delay(200);  **return** 0;  } |

**Coding**

|  |
| --- |
| #include <iobb.h> #define GreenLED 8,16 #define RedLED 8,15 #define Right 8,17 #define Left 8,18 #define DoorStatus2ESP 8,10 #define GarageStatus2ESP 8,9 #define DoorOpen4mESP 8,14 #define DoorClose4mESP 8,11 #define IRbtwnDoor 8,7 #define IRoccupancy 8,8 #define LMcheckDoorOpened 9,30 #define LMcheckDoorClosed 9,12 #define DoorOpen4mArduino 9,25 #define DoorClose4mArduino 9,27 #define Buzzer 9,23 #define PushButton 9,41  void CloseDoor() {  pin\_high(Left);  pin\_low(Right);  while(is\_high(LMcheckDoorClosed))  {  if(!is\_high(IRbtwnDoor))  {  pin\_low(Left);  pin\_low(Right);  pin\_high(Buzzer);  }  else  {  pin\_high(Left);  pin\_low(Right);  }   if(is\_low(PushButton))pin\_low(Buzzer);  }  pin\_low(Left);  pin\_low(Right);  pin\_high(GreenLED);  pin\_low(RedLED);  pin\_low(Buzzer);  pin\_low(DoorStatus2ESP); } void OpenDoor() {  pin\_high(Right);  pin\_low(Left);  pin\_high(Buzzer);  while(is\_high(LMcheckDoorOpened))  {  if(!is\_high(IRbtwnDoor))  {  pin\_low(Left);  pin\_low(Right);  pin\_high(Buzzer);  }  else  {  pin\_high(Right);  pin\_low(Left);  }   if(is\_low(PushButton))pin\_low(Buzzer);   }   pin\_low(Right);  pin\_low(Left);  pin\_high(RedLED);  pin\_low(GreenLED);  pin\_low(Buzzer);  pin\_high(DoorStatus2ESP); }  int main(void) {   iolib\_init();  iolib\_setdir(GreenLED, DigitalOut);  iolib\_setdir(RedLED, DigitalOut);  iolib\_setdir(Right, DigitalOut);  iolib\_setdir(Left, DigitalOut);  iolib\_setdir(DoorStatus2ESP, DigitalOut);  iolib\_setdir(GarageStatus2ESP, DigitalOut);  iolib\_setdir(Buzzer, DigitalOut);  iolib\_setdir(DoorOpen4mESP, DigitalIn);  iolib\_setdir(DoorClose4mESP, DigitalIn);  iolib\_setdir(IRbtwnDoor, DigitalIn);  iolib\_setdir(IRoccupancy, DigitalIn);  iolib\_setdir(LMcheckDoorOpened, DigitalIn);  iolib\_setdir(LMcheckDoorClosed, DigitalIn);  iolib\_setdir(DoorOpen4mArduino, DigitalIn);  iolib\_setdir(DoorClose4mArduino, DigitalIn);  iolib\_setdir(PushButton, DigitalIn); iolib\_delay\_ms(500); //CloseDoor();  while(1) {  if(is\_high(DoorOpen4mArduino) || is\_high(DoorOpen4mESP)) {  OpenDoor(); } else if(is\_high(DoorClose4mArduino) || is\_high(DoorClose4mESP)) {  CloseDoor(); }  if(is\_low(IRoccupancy)) pin\_high(GarageStatus2ESP); else pin\_low(GarageStatus2ESP);  } iolib\_free(); return(0); } |

**Autorun Program:**

1. Find the location of object file of Garagedoor using command pwd
2. I get location as home/debian/GarageDoor
3. Run command - sudo nano /lib/systemd/system/autostart.service
4. Add this to nano terminal --scripted file for autorun

|  |
| --- |
| [Unit] Description=Service File to start the GarageDoor File After=syslog.target network.target [Service] Type=simple ExecStart=/home/debian/GarageDoor [Install] WantedBy=user.target |

Ctrl+o enter ctrl+X

5 run command- sudo systemctl enable autostart.service

6. Sudo reboot

**Final Model**

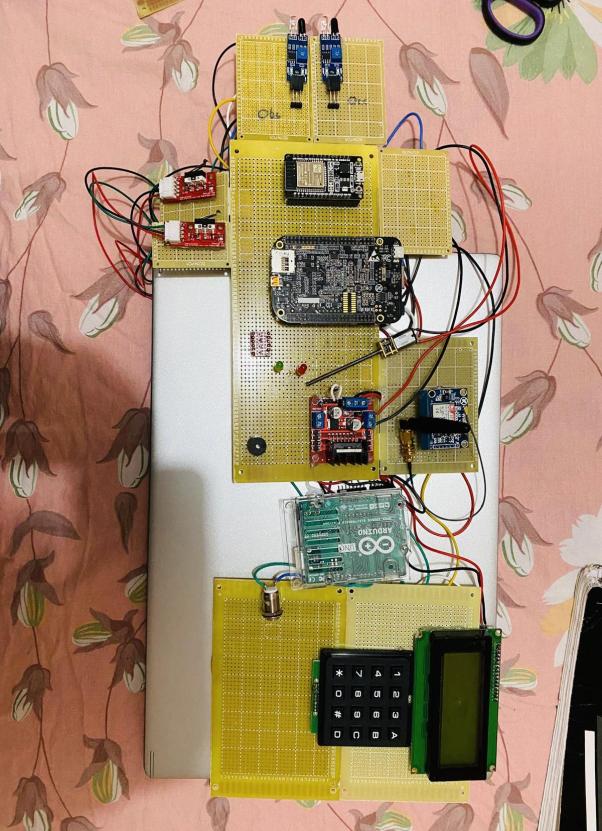
****

Fig.5.13 *Connections of Hardware components*

****

Fig.5.14 *Backview of the project*

****

Fig.5.15 *Final model of our project*

**Output Link:**

<https://www.youtube.com/watch?v=AJK3sm6TUy4>

**Chapter VI**

**Evaluation**

**Introduction**

This chapter will discuss the various issues while selecting the hardware components and connecting them with other components for transferring the data or getting the output. Finally, the chapter will go on discussing methods of solving discrepancies or issues related to Beaglebone Black, IR Sensor, GSM Module and Cloud Service (ThingSpeak).

**Minimum Requirements**

Minimum project requirements will be running the IOT based smart garage door system on Linux and getting the output. Also, another task is to inform the owner through GSM module if anyone tries to open the garage through keypad by entering the wrong password more than three times. While updating the user about the activity via GSM based SMS the system must be capable to get the data updated to the cloud service. As the last requirement the system must be running without the help of any laptop or other devices.

**Troubleshooting**

This section of the chapter will tell you about the various we faced during the entire course of the project. The issues and their solutions are as follows:

* Getting the hardware ready was a problem as some of the GSM may or may not be compatible with the Beaglebone and since this was the first time, we were using GSM thus we had some confusion. To resolve this issue, we had to search many issues and thus after looking at the links we found that either SIM900 or SIM800 would be compatible thus we used SIM800.
* While cloning a git repository for GPIO library. I forget to connect the BBB to the internet via ethernet cable.
* While writing c code. The output is shown as continuous flow on the terminal.
* Use POSIX API library to add delay during suspending operations
* The LED was not glowing at first as I connected it to pin 11 of header P8 but after changing it to pin 12 of header P9 it started glowing.
* In c programming while defining LED pins I defined it as “*#define LEDpin 12,8”* i got error “*segmentation fault*”. then I changed it to “*#define LEDpin 8,12”*
* When I ran my code on bbb the DC motor did not start. Because the connections were loose. I made the connections again and then it ran successfully.
* While writing the code It was difficult for me to understand the logic of the IR sensor. Hence when I initially wrote the code the logic was opposite. But eventually it turned out as it was meant to be.
* Another issue we faced related to the cloud service, whenever we were checking that the data was getting up to the cloud but was not unable to get it displayed then we checked that we had few spelling errors and after that we corrected all the spellings and we could see the data on the cloud.

**Chapter VII**

**Conclusion**

**Future Work**

* We cannot reset the password directly from the keypad, for that we need to do modifications in the code.
* Also, in future use of a camera can also enhance the security feature, like by capturing the vehicle number’s pictures we can save them on the cloud for future use if something happens.
* We design this project only for single garage door, we cannot use this project for multiple garage doors.
* For the wifi connection, we have to add the network credentials in the code, we cannot use any random network for connecting to the cloud.
* This project lacks the feature of LCD lock, like if someone enters the incorrect password more than three times then, automatically the LCD screen will be locked.

**Chapter VIII**

**User’s Guide**

**About the Product**

The product is a smart garage door monitoring system which can be a great feature to your normal garage doors. The system will provide various services to you like accessing the garage door both through webpage as well as with keypad and LCD. Also the information of opening and closing of the garage door will be stored on cloud service called Thingspeak. As a security feature the notification of opening and closing of the garage door is being sent to the user and if anyone enters the wrong password more than 3 times then also the user will get notified.

**Powering Up the System**

To power up the system you will require a power adaptor of 5V, 2A. The power adaptor will be used to power up the Beaglebone Black which is the main microcontroller unit of the product. When it gets powered ON the top LEDs must start blinking in a repeated motion. The Beaglebone Black will provide power to all the other components of the product.



Fig.8.1. *Beaglebone with Adapter*

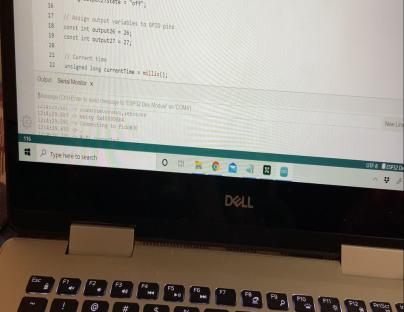
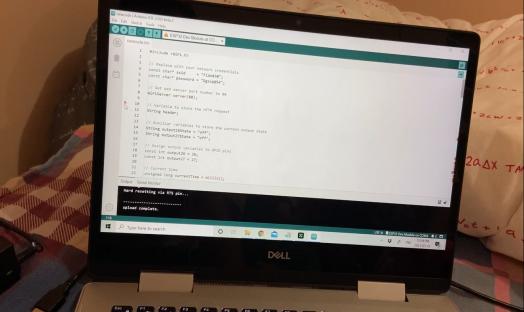
**Connecting Wifi**

When using the product for the first time we need to connect it to the preferred wifi network. To provide wifi service we are using ESP32 wifi module which provides the internet service to the product.

Make sure that you have a strong and preferable wifi network.

To connect to the wifi network make sure you provide the correct SSID and Wifi password into the code to upload to ESP32 wifi module.

*(a)* *(b)*

****

*(c)* *(d)*

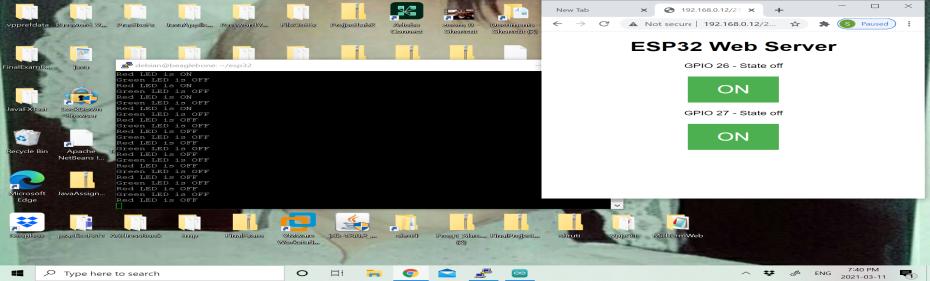
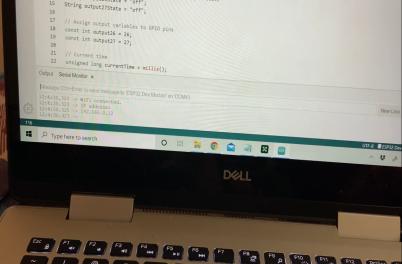
****

Fig.8.2(a) *Uploading Code to Wifi Module; (b) Connecting Wifi to the Wifi Module; (c) Wifi Getting Connected; (d) ESP32 Web Server*

**Connecting to Web Platform**

We are using Thingspeak.com as a cloud service platform.

* The link to the thingspeak account is: <https://thingspeak.com/channels/1320865/>.
* This link can be shared among the users to get the live updates of the garage door status and how many times the door has been accessed.
* The main features of this cloud service are:  
  a) It can show the live status of the door from anywhere.  
  b) The status is in the graph format in three different charts which represents how many times the door has been opened or closed and the status of the garage whether it is vacant or occupied.  
  c) It represents the data very clearly and specifically.

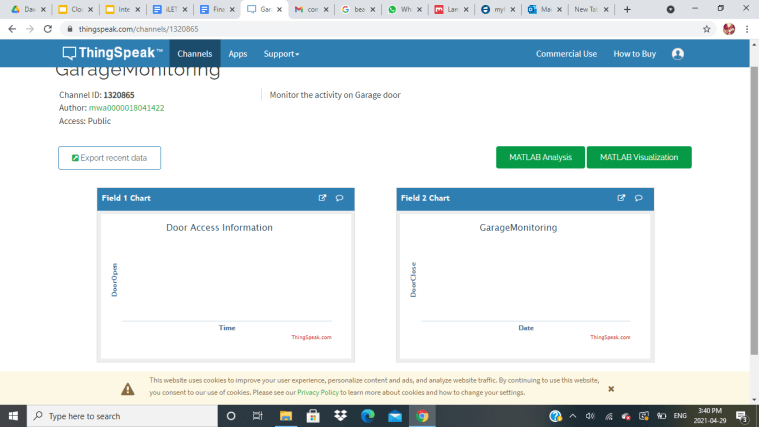


Fig 8.3. *Cloud Service Thingspeak Page*

**Manual Operation of Garage Door**

This product can also be accessed manually as it has been provided with keypad and LCD access as well. When the power has been provided to the LCD screen it will show “Welcome to Garage Monitoring System” and then it will ask you to “Enter Password”. Then you can use the keypad to enter the password which is “1234”. As you will enter the correct password the garage door will automatically open.

To close the door just press the “D” key on the keypad and the door will automatically get closed.

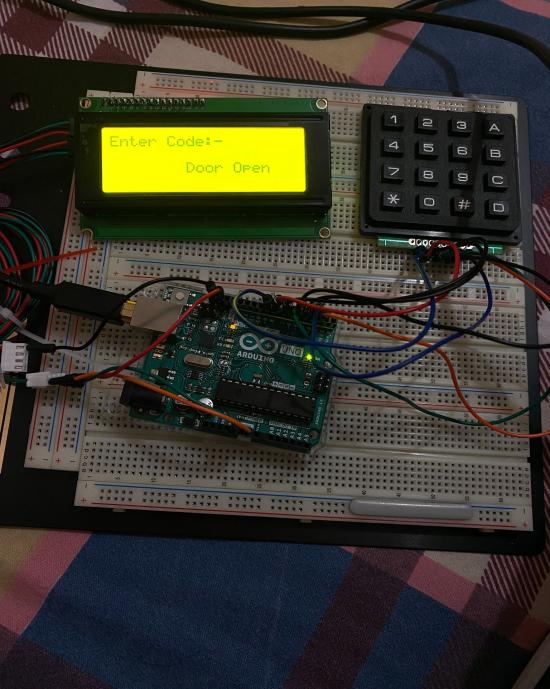


Fig.8.4. *Keypad and LCD*

**Accessing Pushbutton**

There is one pushbutton with the product. The use of the pushbutton is to disable the buzzer. The buzzer provided in the product will get enabled and starts making noise whenever any obstacle comes in between the door while closing it. So you can just press the push button to disable the buzzer.

**References**

1. Retrieved from IoTDesignPro (2019, Nov 07) *IoT Wi-Fi Garage Door Opener using NodeMCU ESP12E*<https://iotdesignpro.com/projects/iot-based-garage-door-opener>
2. Retrieved from PROJECT HUB (2018, Oct 27) *RFID and Keypad Based Door Lock Using Arduino* [https://create.arduino.cc/projecthub/muhammad-aqib/rfid-and-keypad-based-door-lock-using-arduino-89e1d5?ref=tag&ref\_id=home%20security&offse=6](https://create.arduino.cc/projecthub/muhammad-aqib/rfid-and-keypad-based-door-lock-using-arduino-89e1d5?ref=tag&ref_id=home%20security&offset=6)
3. Retrieved from *Robu.in*, Tuesday May19,2020, Priyanka D. URL: <https://robu.in/ir-sensor-working/>
4. Retrieved from *IndustrialShields.com*, 2020, URL:<https://www.industrialshields.com/blog/arduino-industrial-1/post/how-to-use-the-gprs-module-sim800l-library-129>
5. Retrieved from *Cuidevices.com, basics technologies and buzzers and tone driving circuits,* Bruce R,2021 URL:<https://www.cuidevices.com/blog/buzzer-basics-technologies-tones-and-driving-circuits>
6. Retrieved from *Lastminuteengineers.com, Arduino keypad tutorial*, 2021, URL: <https://lastminuteengineers.com/arduino-keypad-tutorial/>
7. Retrieved from *lastminuteengineers.com L298N with DC Motor* ,2021 <https://lastminuteengineers.com/l298n-dc-stepper-driver-arduino-tutorial/>
8. Retrieved from *thomas.net, Instruments control limit switche*s, 2021 URL: <https://www.thomasnet.com/articles/instruments-controls/limit-switches/>
9. Retrieved from *Instructables.com, interfacing lcd 20\*4 display with Arduino*,URL: <https://www.instructables.com/Interfacing-20x4-LCD-with-Arduino/>
10. Retrieved from *Industrialshields.com, module esp32,* 2020, URL: <https://www.industrialshields.com/blog/arduino-industrial-1/post/wifi-bluetooth-module-esp32-133>
11. Retrieved from *Cuidevices.com, basics technologies and buzzers and tone driving circuits,* Bruce R,2021 URL:<https://www.cuidevices.com/blog/buzzer-basics-technologies-tones-and-driving-circuits>
12. Retrieved from TEPTechnopreneur (2016, Nov 13) *6 Necessary Embedded Tools* URL:[https://www.theengineeringprojects.com/2016/11/necessary-embedded-tools.](https://www.theengineeringprojects.com/2016/11/necessary-embedded-tools.html#:~:text=These%20embedded%20tools%20include%20editors%2C%20compilers%2C%20assemblers%2C%20debuggers%2C,the%20embedded%20tools%20are%20necessary%20and%20work%20together)
13. Retrieved from *educba.com, What is eclipse IDE?* URL:<https://www.educba.com/what-is-eclipse-ide/>
14. Retrieved from *filehippo.com*(2019, Jan 30) URL:<https://filehippo.com/download_tera_term/?ex=CORE-190.2>
15. Retrieved from *github.com* (2020, Sept 24) URL:<https://github.com/arduino/arduino-pro-ide>
16. Retrieved from *geeksforgeeks.org* (2019, Aug 26) *Difference between Compiler and assembler* URL:<https://www.geeksforgeeks.org/difference-between-compiler-and-assembler/>
17. Retrieved from *geeksforgeeks.org*(2019, Nov 21) *Linker* URL:<https://www.geeksforgeeks.org/linker/>
18. Retrieved from *electronics-notes.com, USB operation: Protocol, Data transfer & packets* <https://www.electronics-notes.com/articles/connectivity/usb-universal-serial-bus/protocol-data-transfer.php>
19. Retrieved from *teachmemicro.com*, *BEAGLEBONE BLACK SERIAL CONNECTION TO ARDUINO*<https://www.teachmemicro.com/beaglebone-black-serial-arduino/>
20. Retrieved from *Trevors Electronic Blog*, *How to get serial UART port working on the beaglebone black.* <https://electronics.trev.id.au/2018/02/09/get-uart-serial-ports-working-beaglebone-black/>
21. Retrieved from *electronics-lab.com, Using a 20\*4 I2C Character LCD With Arduino UNO* <https://www.electronics-lab.com/project/using-20x4-i2c-character-lcd-display-with-arduino-uno/>
22. Retrieved from *robu.in*(2020, Jan 9), *Interfacing Sim 800L with Arduino.* <https://robu.in/sim800l-interfacing-with-arduino/>
23. Retrieved from *github.com*(2020, Nov 17), *Arduino-SIM800L- driver* <https://github.com/ostaquet/Arduino-SIM800L-driver>
24. Retrieved from *electronicwings.com*, *4\*4 Keypad interfacing with Arduino Uno* <https://www.electronicwings.com/arduino/4x4-keypad-interfacing-with-arduino-uno>
25. Retrieved from *instructables.com, Connect ESP32 to AskSensors over https,* <https://www.instructables.com/How-to-Connect-an-ESP32-to-the-IoT-Cloud/>
26. Retrieved from *derekmolloy.ie*, *Writing a Kernel Linux module- Part 3: Buttons and LED* URL: <http://derekmolloy.ie/kernel-gpio-programming-buttons-and-leds/>
27. Retrieved from beagleboard.org (2016,June 5) URL: <https://beagleboard.org/p/hoffmanjon/controlling-a-motor-with-an-h-bridge-fd13b2>
28. Retrieved from, learn.adafruit.com, Using an ir sensor URL:https://learn.adafruit.com/ir-sensor/using-an-ir-sensor
29. Retrieved from instructables.com (2018, May 6) *ESP32: Internal Details and Pinout* <https://www.instructables.com/ESP32-Internal-Details-and-Pinout/>
30. Retrieved from, Perforce.com (2021), MISRA C and MISRA C++ <https://www.perforce.com/resources/qac/misra-c-cpp#why>
31. Retrieved from, w3schools.com, HTML Introduction <https://www.w3schools.com/html/html_intro.asp>
32. Retrieved from, educba.com, Versions of HTML <https://www.educba.com/versions-of-html/>
33. Retrieved from *Components101.com, L298N Motor Driver(2020, April13)* <https://components101.com/modules/l293n-motor-driver-module>
34. Retrieved from, ThingSpeak   
    <https://thingspeak.com/>
35. Retrieved from, Code Project, An Introduction to Thingspeak <https://www.codeproject.com/Articles/845538/An-Introduction-to-ThingSpeak>
36. Retrieved from, MDN Web Docs, An Overview of HTTP <https://developer.mozilla.org/en-US/docs/Web/HTTP/Overview>
37. Retrieved from, Webopedia  
    <https://www.webopedia.com/definitions/802-11/>
38. Retrieved from, MQTT.org   
    <https://mqtt.org/>
39. Shabaz (2019,August 14) Element14.com Retrieved from <https://www.element14.com/community/community/designcenter/single-board-computers/next-genbeaglebone/blog/2019/08/15/beaglebone-black-bbb-io-gpio-spi-and-i2c-library-for-c-2019-edition>
40. Retrieved from, randomnerdtutorial.com <https://randomnerdtutorials.com/esp32-web-server-arduino-ide/>
41. Retrieved from IoT Design Pro <https://iotdesignpro.com/projects/how-to-send-data-to-thingspeak-cloud-using-esp32>
42. Arduino lesson- IR obstacle Avoidance sensor, Osoyoo (2017,july 21). Retrieved from <https://osoyoo.com/2017/07/24/arduino-lesson-obstacle-avoidance-sensor/>
43. Usleep Linux manual page (2017,September 15) Retrieved from <https://man7.org/linux/man-pages/man3/usleep.3.html>
44. Monk S. (2021) Blinking an LED with BeagleBone Retrieved from <https://learn.adafruit.com/blinking-an-led-with-beaglebone-black/writing-a-program>
45. Retrieved from, EasyEDA  
    <https://docs.easyeda.com/en/FAQ/Editor/index.html>
46. Retrieved from, twistedtraces.com <https://www.twistedtraces.com/blog/learn-about-different-types-of-pcbs-and-their-advantages>
47. Retrieved from, vse.com

<https://www.vse.com/blog/2019/10/29/gerber-files-explained-understanding-their-role-in-pcb-manufacturing/>