ZAGAZIG UNIVERSITY

Graduation Project Book

Smart Electric Bike Conversion Kit

CSE 23

Rider

APPROVAL SHEET

Smart E-Bike Conversion Kit

Project Team:

Ahmed Ramadan Elsaid Ali

Ahmed Saeed Abdullah Disoky

Ghada Hassan Abd Elmonem Mohamed

Hady Ibraheem Mohamed Ibraheem

Kerolos Ayman Gerges Ayoub

Mariam Mahmoud Elsaid Abdullah

Mohamed Abd Elshafy Ibraheem Mohamed

Mohamed Magdy Mohamed Abd Elsadek

Mohamed Mahmoud Hassan Mostafa

Nourhan Hussein Ibraheem Hassan

Omar Mostafa Gouda Attia

Yasser Fares Hassan Abd Elhameed

Computer and Systems Engineering, Class of 2023

Supervisor: Dr. Sanaa Fekry Abd Elsadek . Assistant Professor

Date Approved: 15/7/2023.

Abstract

This graduation project focuses on designing and developing an electric bike conversion kit, which can transform a regular bicycle into an e-bike. The kit includes a motor, battery, controller, and other necessary components to enable pedal-assisted riding. The project aims to provide an affordable and eco-friendly solution for commuters who prefer cycling but need some assistance to tackle hills or cover longer distances or commute to work daily without overly exhausting themselves for it. The kit was designed to be easily installed on various types of bikes and customizable to fit different user preferences. The project includes extensive testing and optimization to ensure the kit's safety, anti-theft security, reliability, and efficiency. The successful completion of the project demonstrates the potential of such conversion kits to promote sustainable transportation and enhance cycling accessibility and the potential for further optimization and upgrading for more production class consistent product.

Acknowledgements

First, all thanks are due purely to Allah, for providing us the blessings and the strength to complete this project. Second, we would like to express our deepest appreciation to Dr. Sanaa, for her guidance through the whole process and for being so supportive and encouraging, Allah bless you, we can't thank you enough.

All our friends and colleagues for being so supportive and such good friends. We couldn't do it without you. And above all comes the family I'd like to thank them because they believed in us, always gave us confidence, and never got tired of pushing us forward.

Dedication

We dedicate this to our families, our college, our professors, our department students, our community, and all our country.

Table of Contents

Part One: Electric Bike:

Part Two: User Interface:

Chapter 1: Introduction

The basic goal of this attempt is to create a better, more efficient, healthy, eco-friendly, and futuristic mean of transportation as well as it's being a contributing solution for several local economic and environmental problems while adding more ease and accessibility than usual e-bikes in the market utilizing an IoT sensor network implemented into the bike with rest of it's kit with the center point being the Node MCU ESP 32. The sensor network mostly dedicates it's features to user and bike safety and anti-theft security.

1.1: Introduction to Bicycle Mechanics:

The bicycle might be one of the greatest inventions of all time. It is simple, efficient and fun transportation. You can get to places quickly by converting your body's energy into kinetic energy, energy in motion. A bike is a machine that can magnify force or speed to move you up a hill more easily. It is also capable of changing the energy from the human body to energy for motion. Energy in all forms cannot be created or destroyed, energy can only change from one form to another. Biking can feel like hard work at times because you must use a lot of muscle force to pedal. If you are going uphill you must work against the force of gravity. If you are trying to go fast you are working against a force called air resistance. If there are any bumps or turns then you must use energy to slow down and use the force of friction from the brakes. When riding a bike, you are always using energy to make the wheels go around and keep in motion.

1.2: Introduction to Electric Bikes:

An electric bike, also known as an e-bike, is a bicycle that has an electric motor, its controller, and a rechargeable battery. The motor assists the rider's pedaling and can be used to propel the bike forward, making it easier to ride uphill or against headwinds. E-bikes come in a variety of styles, including road bikes, mountain bikes, and commuter bikes.

E-bikes are becoming increasingly popular due to their many benefits. They provide an alternative to traditional modes of transportation that are more environmentally friendly and less expensive than cars as well as being way slimmer than cars which solves traffic issues. They also offer a convenient way to commute that requires minimal effort, making them ideal for people who may not be able to ride a traditional bike due to physical limitations or lack of fitness.

One of the key advantages of e-bikes is that they allow riders to travel further and faster than they would be able to on a traditional bike. The electric motor provides a boost that can help riders maintain speeds of up to 32km/h with ease, and the battery can provide enough power for distances of up to 80km or more, depending on the model and conditions.

There are several types of e-bike motors, including hub motors and mid-drive motors. Hub motors are located in the wheel hub and provide direct power to the wheel, while mid-drive motors are located near the pedals and provide power through the bike's chain.

E-bikes are typically more expensive than traditional bikes, but the cost is offset by the savings on fuel and maintenance over time. They also offer a unique and exciting riding experience that combines the best of traditional biking with the convenience and efficiency of electric power.

1.3: Introduction to Embedded Systems:

An embedded system is a computer system that is designed to perform a specific task or set of tasks. These systems are typically built into other devices or machines, such as appliances, automobiles, medical equipment, and industrial machinery. They are specialized computer systems that are optimized for a particular application and are designed to operate reliably and efficiently in a wide range of environments.

Embedded systems typically consist of a microcontroller or microprocessor, which is the central processing unit (CPU) of the system. The microcontroller is responsible for executing the software that controls the system and manages its input and output. Other components of an embedded system may include sensors, actuators, memory, and communication interfaces.

Embedded systems are used in a wide range of applications, from simple consumer electronics to complex industrial control systems. They can be found in appliances like washing machines, refrigerators, and air conditioners, as well as in automobiles, medical devices, and aerospace applications.

One of the key advantages of embedded systems is their ability to perform specific tasks with high accuracy and reliability. They are also highly efficient and can operate for long periods of time without requiring maintenance or repair. Additionally, their small size and low power consumption make them ideal for use in portable or battery-operated devices.

Designing and developing embedded systems requires specialized knowledge and skills in hardware and software engineering. Embedded systems engineers must have a deep understanding of computer architecture, electronics, programming, and real-time operating systems. They must also be able to work closely with other

engineers and stakeholders to ensure that the system meets the requirements of its intended application.
We will be connecting sensors with each other through communication protocols into a microcontroller to make a sustained and reliable sensor network.

1.4: Introduction to Communication Protocols:

A communication protocol is a set of rules and standards that govern the exchange of data between two or more devices. Communication protocols are essential for ensuring that devices can communicate with each other reliably and efficiently. They define how data is transmitted, formatted, and interpreted, and they provide a common language that devices can use to understand each other.
There are many different types of communication protocols, including:

Serial protocols: These protocols transmit data one bit at a time over a single wire or channel. Examples include RS-232, SPI, and I2C.

Parallel protocols: These protocols transmit data across multiple wires or channels simultaneously. Examples include SCSI and PCI.

Wireless protocols: These protocols transmit data over the airwaves using radio waves or other wireless signals. Examples include Wi-Fi, Bluetooth, and Zigbee.

Network protocols: These protocols govern the communication between devices on a network, such as the internet. Examples include TCP/IP, HTTP, and FTP.

Fieldbus protocols: These protocols are used in industrial automation and control systems to communicate with sensors, actuators, and other devices on a factory floor. Examples include Modbus, CAN, and Profibus.

Each communication protocol has its own strengths and weaknesses, and the choice of protocol will depend on factors such as the type of devices being used, the distance between devices, the required speed and reliability of communication, and the environment in which the devices are operating.

Communication protocols are typically implemented in software or firmware, and they are often standardized by industry organizations or standards bodies to ensure interoperability between devices from different manufacturers.

1.5: Introduction to IoT using NodeMCU:

The Internet of Things (IoT) refers to the connection of everyday objects and devices to the internet, allowing them to send and receive data and interact with other devices and services. One popular platform for building IoT devices is NodeMCU, which is an open-source firmware based on the ESP8266/ESP32 Wi-Fi module.

NodeMCU allows developers to easily connect sensors, actuators, and other components to the internet and interact with them using a variety of programming languages and tools. Some of the key features of NodeMCU include:
Easy programming: NodeMCU uses Lua scripting language, which is easy to learn and use, even for beginners.

Built-in Wi-Fi: NodeMCU has a built-in Wi-Fi module, which allows it to connect to the internet and communicate with other devices.

GPIO pins: NodeMCU has several general-purpose input/output (GPIO) pins, which can be used to connect sensors, actuators, and other components.
Small size: NodeMCU is small in size, making it ideal for use in small-scale IoT projects.
Some examples of IoT projects that can be built with NodeMCU include:

- Smart home devices: NodeMCU can be used to build smart home devices, such as thermostats, lighting controls, and security systems.
- Environmental monitoring: NodeMCU can be used to monitor environmental conditions, such as temperature, humidity, and air quality.
- Industrial automation: NodeMCU can be used to control and monitor industrial machinery and equipment.

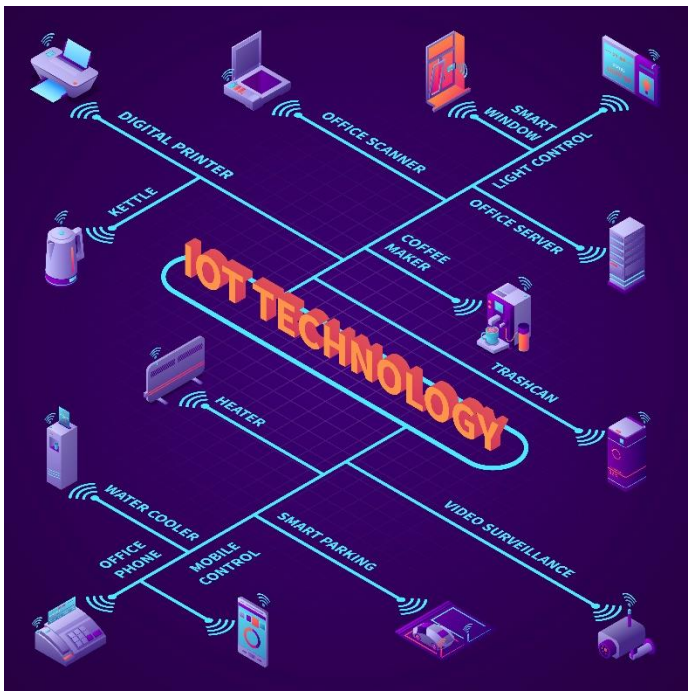


Figure 1.5.1

1.6: Introduction to Mobile Application:

A mobile application, or "app" for short, is a software program designed to run on mobile devices such as smartphones and tablets. Mobile apps are typically downloaded from an app store, such as the Apple App Store or Google Play, and installed on the user's device.

Mobile apps can be designed for a variety of purposes, from entertainment and social networking to productivity and business. They can be used to access information, perform tasks, play games, communicate with others, and much more.

Mobile apps are typically developed using programming languages such as Java, Swift, or Kotlin, and they are designed to be optimized for the small screens and limited processing power of mobile devices. They may also use features such as touchscreens, cameras, GPS, and other sensors to provide unique and engaging user experiences. We have designed our application from scratch using flutter and figma for UI/UX design.

There are two main types of mobile apps: native apps and web apps. Native apps are designed specifically for a particular mobile platform, such as iOS or Android, and are installed directly on the user's device. They have access to the device's hardware and software features, making them more powerful and responsive than web apps. Web apps, on the other hand, are essentially mobile-optimized websites that are accessed through a mobile browser. While web apps can be more easily developed and maintained than native apps, they may not have access to all of the device's features, and may not be as responsive or seamless as native apps.

Mobile apps have become an increasingly important part of modern life, and they are used by millions of people around the world every day. They have transformed the way we interact with technology, and they have created new opportunities for businesses, developers, and entrepreneurs to reach audiences and provide value in new and innovative ways.

1.7: Introduction to E-commerce Website:

An e-commerce website is a virtual storefront that allows businesses to sell products or services online. E-commerce websites provide customers with the ability to browse products, make purchases, and complete transactions, all from the comfort of their own homes or mobile devices.

E-commerce websites can be designed to sell physical products, digital products, or both. They can be used by businesses of all sizes, from small startups to large multinational corporations. E-commerce websites typically include features such as product listings, shopping carts, payment gateways, and order tracking. We have created an e-commerce website in order to add realism to the project as a viable business product that can be sold and marketed using modern ways and ideas.

One of the key advantages of e-commerce websites is their ability to reach a global audience. With an e-commerce website, businesses can sell their products or services to customers anywhere in the world, 24 hours a day, seven days a week. E-commerce websites also provide customers with the ability to compare prices and products from a wide range of businesses, making it easier to find the best deals and the products that meet their needs.

To create an e-commerce website, businesses typically use an e-commerce platform or content management system (CMS) that provides the necessary features and functionality. Popular e-commerce platforms include Shopify, WooCommerce, and Magento, among others. These platforms provide businesses with tools to manage their products, track inventory, process payments, and more.

Designing an e-commerce website requires careful consideration of factors such as user experience, branding, and security. The website should be easy to navigate, visually appealing, and optimized for search engines to attract customers. It should also be secure, with measures such as SSL encryption and two-factor authentication to protect customer data and transactions.

Figure 1.7.1

Part One: Electric Bike

Chapter 2: Primary Components

The **main** components used to convert any normal bicycle to an electric one are as follows:

- ❖ BLDC Hub Motor (Rear/Front)
- ❖ Motor Controller (PWM + Extra Functions)
- ❖ Li-ion/Li-Po Battery (48V+)

We shall provide a simple explanation to each of them and how they work and why are they used in our project (how do they contribute to the making of an E-bike)

Electric bikes usually have two main types of assisting methods: Pedal assist and Throttle, some use only one type some use both, we have used both in this project to maximize flexibility and acceptance of user preference.

A simple Block Diagram representation of the main network of components and how the controller controls their inputs/outputs:

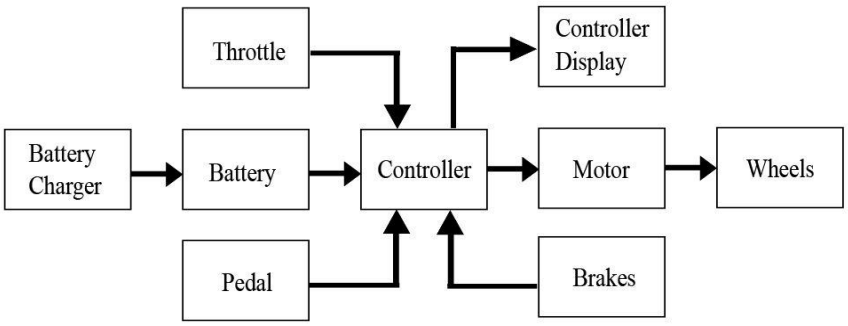


Figure 2.0.1

2.1: Electric BLDC Motor:

2.1.1: Functionality

A Brushless DC (BLDC) motor is an electric motor that uses electronic commutation instead of brushes to control the movement of the rotor.

BLDC motors are more efficient and reliable than traditional brushed DC motors, making them a popular choice for a wide range of industrial and consumer applications.

The basic functionality of a BLDC motor is similar to that of a traditional DC motor. The motor consists of a stator, which contains the stationary coils, and a rotor, which contains the permanent magnets. When an electrical current is applied to the coils, a magnetic field is generated that interacts with the magnetic field of the rotor, causing the rotor to spin.

In a BLDC motor, the commutation is achieved using electronic circuits instead of brushes. The electronic commutation system uses sensors to detect the position of the rotor and then switches the current in the coils to maintain the correct alignment between the stator and rotor magnetic fields. This allows the motor to operate more efficiently and with less wear and tear on the moving parts.

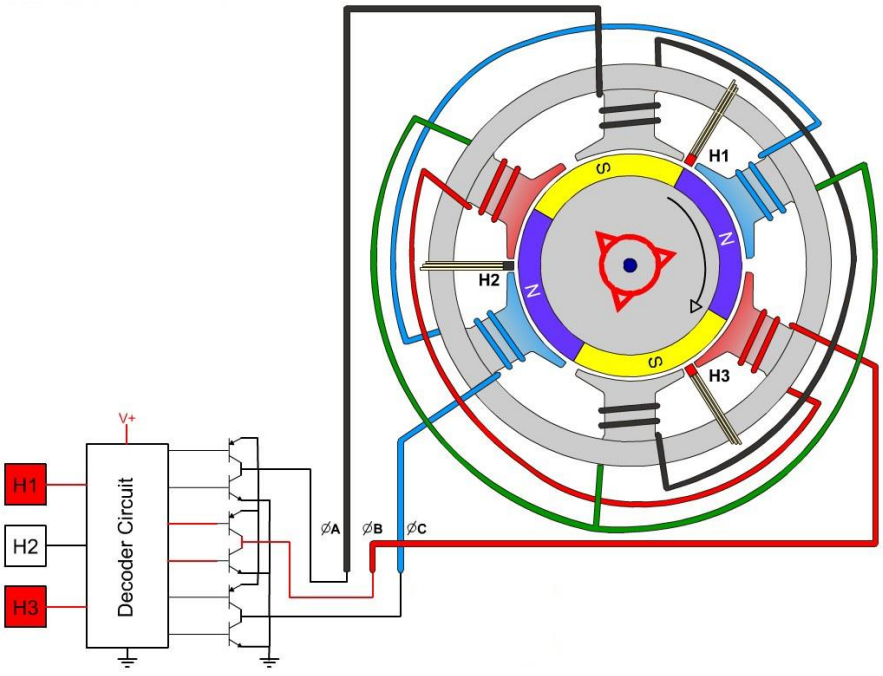


Figure 2.1.1.1

BLDC motors have several advantages over traditional DC motors. They are more efficient, with higher power-to-weight ratios and lower heat generation. They also have a longer lifespan, as there are no brushes to wear out or create friction. Additionally, BLDC motors can operate at higher speeds and have better control over torque and speed, making them ideal for applications that require precise control over motor performance like our project.

2.1.2: Working Principle

The working principle of a Brushless DC (BLDC) motor is based on the interaction between magnetic fields created by the stator and the rotor. Unlike traditional brushed DC motors, BLDC motors use electronic commutation to control the direction of current flow through the motor's windings.

The BLDC motor consists of a stator, which contains the stationary coils, and a rotor, which contains permanent magnets. When an electrical current is applied to the coils in the stator, a magnetic field is generated that interacts with the magnetic field of the rotor, causing the rotor to turn.

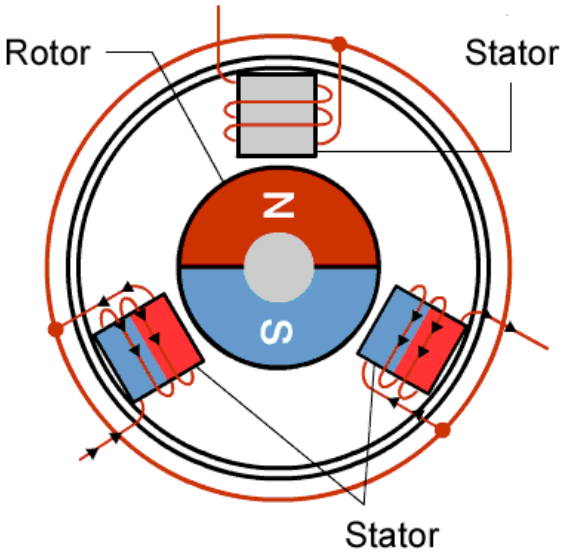


Figure 2.1.2.1

To control the direction of current flow through the stator coils, BLDC motors use electronic commutation. Electronic commutation is achieved using sensors that detect the position of the rotor and then switch the current in the coils to maintain the correct alignment between the stator and rotor magnetic fields. There are three commonly used methods for electronic commutation in BLDC motors:

- Hall effect sensors: Hall effect sensors are mounted on the stator and detect the position of the rotor using the magnetic field of the rotor. The sensors then send signals to the motor controller to switch the current in the coils at the appropriate time.
- Back electromotive force (EMF) sensing: Back EMF sensing uses the voltage generated in the coils by the moving rotor to detect the position of the rotor. The motor controller uses this information to switch the current in the coils at the correct time.
- Sensorless commutation: Sensorless commutation uses algorithms to estimate the position of the rotor based on the motor's back EMF, without the need for dedicated position sensors.

But mainly the hall effect sensor method is the most common and also the one used in this project’s motor.

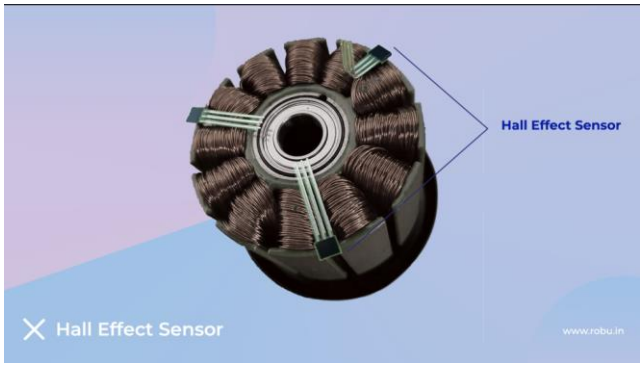


Figure 2.1.2.2

2.1.3: Mid-drive vs Hub-drive motor

Electric bicycles (e-bikes) can be powered by either hub motors or mid-drive motors. Both types of motors have advantages and disadvantages, and the choice between them depends on the rider's needs and preferences.

Hub motors are located in the hub of the front or rear wheel and provide direct drive to the wheel. Hub motors are generally simpler and less expensive than mid-drive motors, and they are easier to install or retrofit to an existing bike. They also provide a smooth and quiet ride, as there are no gears or chains to shift.

However, hub motors have some drawbacks. They can add significant weight to the bike, as the motor, battery, and controller are all located in the wheel hub. This can affect the handling and balance of the bike, especially at high speeds. Hub motors are also less efficient than mid-drive motors, as they do not take advantage of the bike's gears to provide torque and speed.

Mid-drive motors are located in the center of the bike, near the bottom bracket, and provide power directly to the bike's chain. Mid-drive motors are more efficient than hub motors, as they can take advantage of the bike's gears to provide torque and speed. This makes them ideal for steep hills or off-road terrain, where the rider needs extra power and control.

Mid-drive motors are also more balanced and stable than hub motors, as the weight of the motor and battery is centered on the bike. This makes them better suited for high-speed riding or long-distance touring. However, mid-drive motors are generally more expensive and complex than hub motors, and they can be more difficult to install or retrofit to an existing bike, as well as they aren’t available in the local market. We chose the hub motor as our driving motor for our project as

it's easier to install and control, also it's inner mechanism is relatively easier to comprehend than mid-drives as they are a relatively new and revolutionary option.

Figure 2.1.3.1

2.1.4: Hub-drive Motor Internal Structure

The internal structure of a Brushless DC (BLDC) motor consists of several key components that work together to generate rotary motion. These components include the stator, rotor, bearings, and electronic controller.

The stator is the stationary part of the motor and contains the coils that generate the magnetic field. The stator is typically made of laminated steel sheets that are stacked together to reduce magnetic losses. The coils are wound around the stator teeth, which are the protruding parts of the stator that interact with the rotor.

The rotor is the rotating part of the motor and contains the permanent magnets that interact with the stator's magnetic field. The rotor can be either an internal rotor or an external rotor. In an internal rotor design, the permanent magnets are attached to the inside of the rotor, and the rotor spins on the inside of the stator. In an external

rotor design, the permanent magnets are attached to the outside of the rotor, and the rotor spins on the outside of the stator.

The bearings are used to support the rotor and allow it to spin freely. The bearings are typically made of steel and are located at both ends of the motor. The bearings can be either ball bearings or sleeve bearings, depending on the motor's design and application.

The electronic controller is used to control the motor's speed and direction. The controller uses feedback from sensors, such as Hall effect sensors or back EMF sensors, to determine the position of the rotor and then switches the current in the coils to maintain the correct alignment between the stator and rotor magnetic fields. The controller can also adjust the current flow to the coils to control the motor's speed and torque.

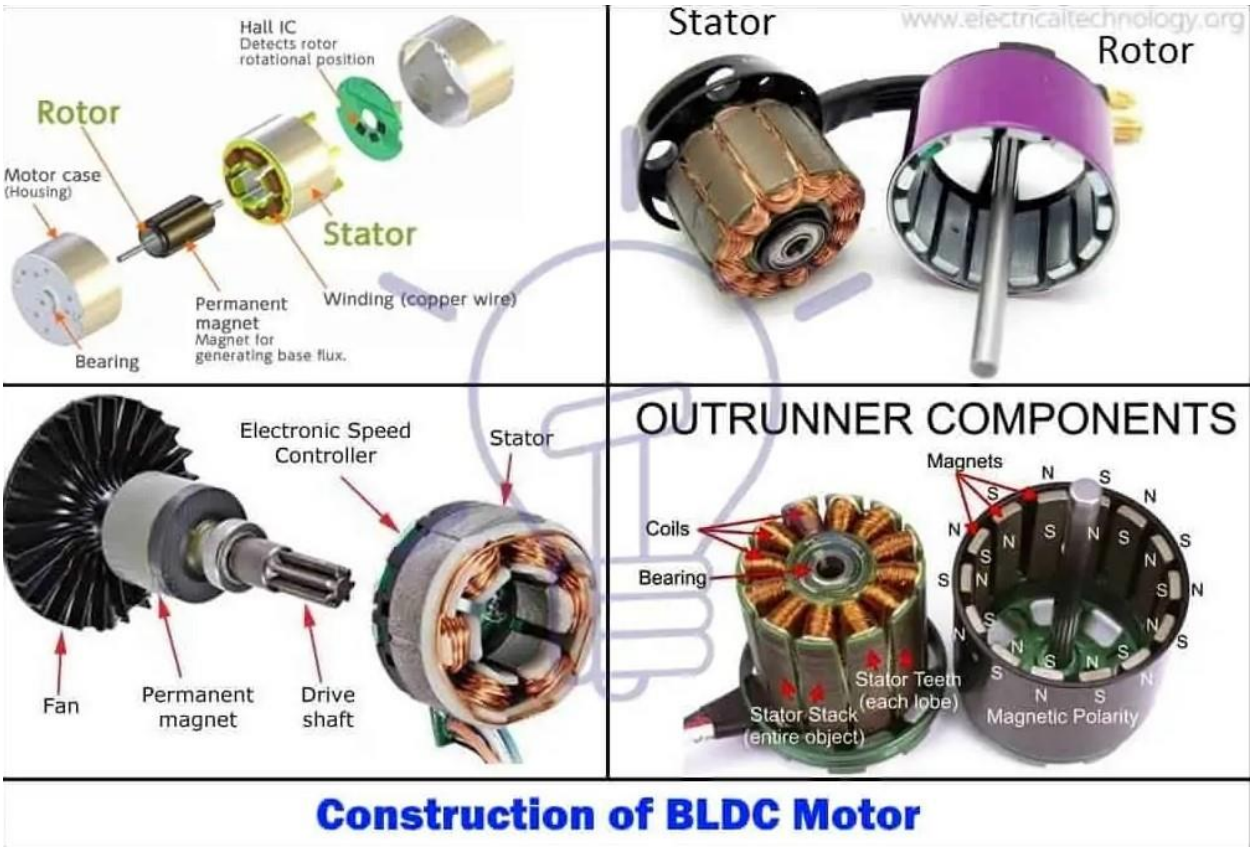


Figure 2.1.4.1

2.2: E-bike Controller:

2.2.1: Functionalities

What is the E-bike controller?

The motor controller is also known as the electric e-bike controller or electric speed controller. It is a circuit board in a sealed protective box with several connection wires sticking out.

It's mounted inside of an e-bike connecting and putting all key components together, like motor, battery, paddle assistant, LCD display, sensors, and throttle, to control the motor's speed, start, and stop.

If the motor acts as the "heart" of an e-bike, then the controller is their "brain," which is the main factor to determine the performance of an e-bike.

The controller serves two main purposes:

- It controls and regulates the flow of current from the battery to the motor in an e-Bike.
- Most controllers have the capability to provide regenerative braking. The resting power is going to be sent back from the motor to the battery to save energy when it works.

Figure 2.2.1.1

Voltage and Power of E-Bike motor controller

The higher current and voltage the controller has, the more powerful eBike it can drive. 36V, 48V, and 60V are the most common controllers for electric scooters and e-bikes among the common kinds from 12 volts to 72 volts.

High-performance eBike are usually equipped with high voltage controllers, such as 52V, 25A, 60V 40A, and 100V, 400A.

Most commute scooters or eBike just have a single controller while extreme performance versions typically have dual motors to get more power.

Controller Display

Figure 2.2.1.2

The controller display is usually located in the middle of the handlebar of the eBike for viewing convenience.

This LCD screen is the key interface between you and your eBike.

It can turn the motor on and off and keep you up to date on speed, where you are in the total range, battery level, pedal assist level, and more

And you can operate on its menu to set up the power of assist, and to know if you need to charge your battery.

How does the controller works

The motor, as we know, translates the electrical energy into mechanical energy, and the controller is the bridge that takes power from the battery and delivery it to the motor depending on the sensors and rider's input.

As the central hub, the controller receives signals from the throttle regulated by the users then give instructions on how much power is going to be sent to the motor to control the speed of the e-bike.

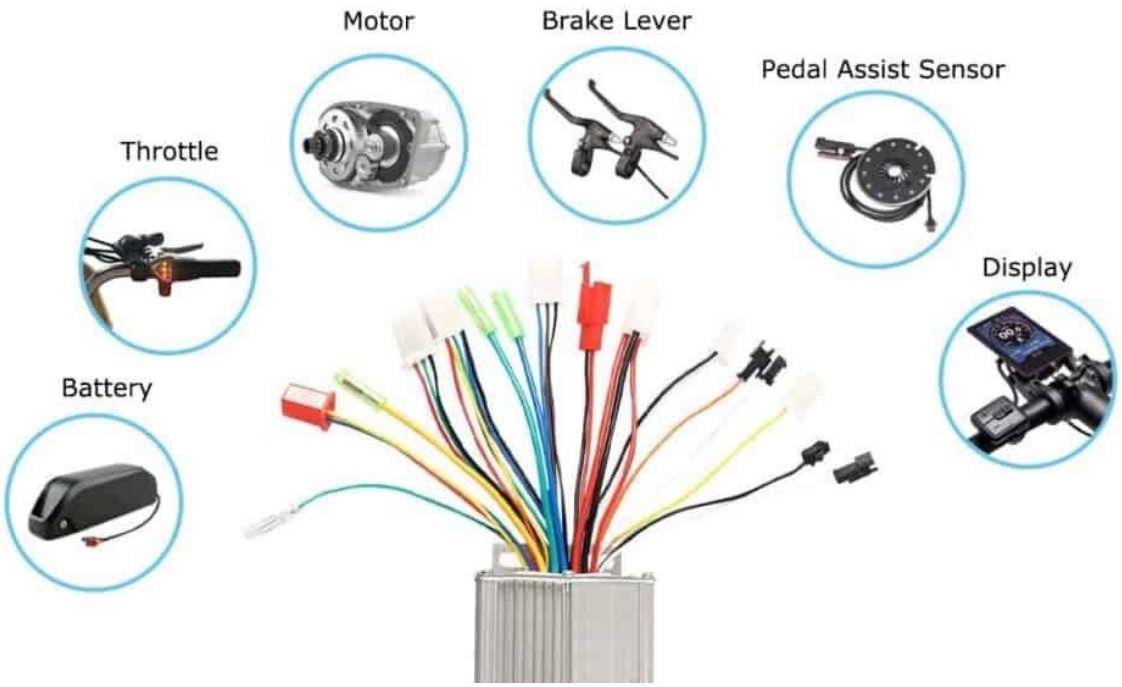


Figure 2.2.1.3

The principle behind the controller is simple:

The motor controller is made of a processor that receives orders or requests signals from the battery, motor speed sensor, or brake sensor and gives a correct output signal such as to set the motor running at a specific speed.

The hall sensors on the circuit board of the controller can monitor and control the motor speed.

What does the microprocessor do?

A microprocessor is rather like a computer running firmware. It sends weak signals to the FETs network, which in turn controls and drives the motor. Usually, the processor forces the FETs to switch on and off quickly by using a high-frequency signal, which makes the controller spin slower or faster

The processor or microchips can also control the timing of power delivery for the motor which makes sense at it is just how brushless DC motor works.

How does the processor affect the motor's performance

More and more brushless DC motors are being used nowadays instead of brushed motors.

To know how the process affects the motor's performance, let's get into a shallow knowledge of brushless DC motors first

Brushless DC motor

For brushless DC motor, there are permanent magnets on the rotor and electromagnets are on the stator

Permanent magnets are usually made of hard materials like steel that has and show their own magnetic field forever, while electromagnet is the coil of the iron core that only release its magnetic when an electric current pass through it.

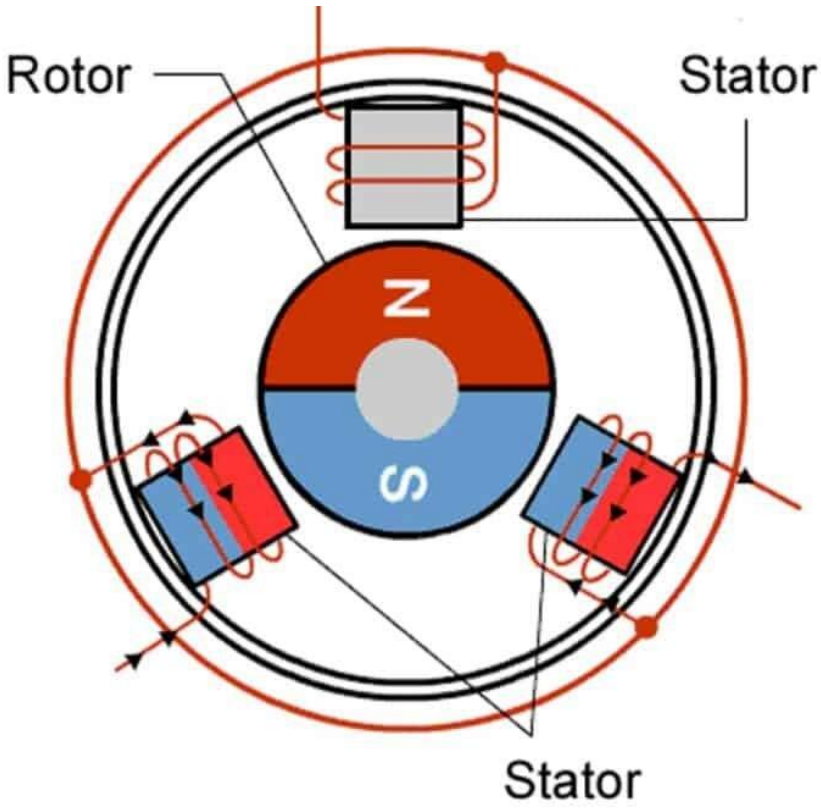


Figure 2.2.1.4

So in order to make the motor work, you need to make the permanent magnet rotate, and to control the rotation, you need to adjust the current into the coils on these fixed electromagnets around.

And processor will do the work to active the electromagnets with precise timing depending on the rotation speed to control the timing of power delivery.

What are the functions of an electric bike controller?

The controller accepts all inputs from the motor, battery, paddle assistant, LCD display, sensors.

The controller gives feedback depending on the above inputs to control every aspect of the e-bike. Things like speed of the motor, amount of pedal assistance, and the brake system.

The controller monitors acceleration, speed, power, voltage, pedaling level, and more

The controller protects inner circuit systems, such as over-voltage and current protection, low-voltage protection, and overheating protection.

2.2.2: Types of Controllers

Different types of E-bike controller

Controllers can be classified according to the following criteria:

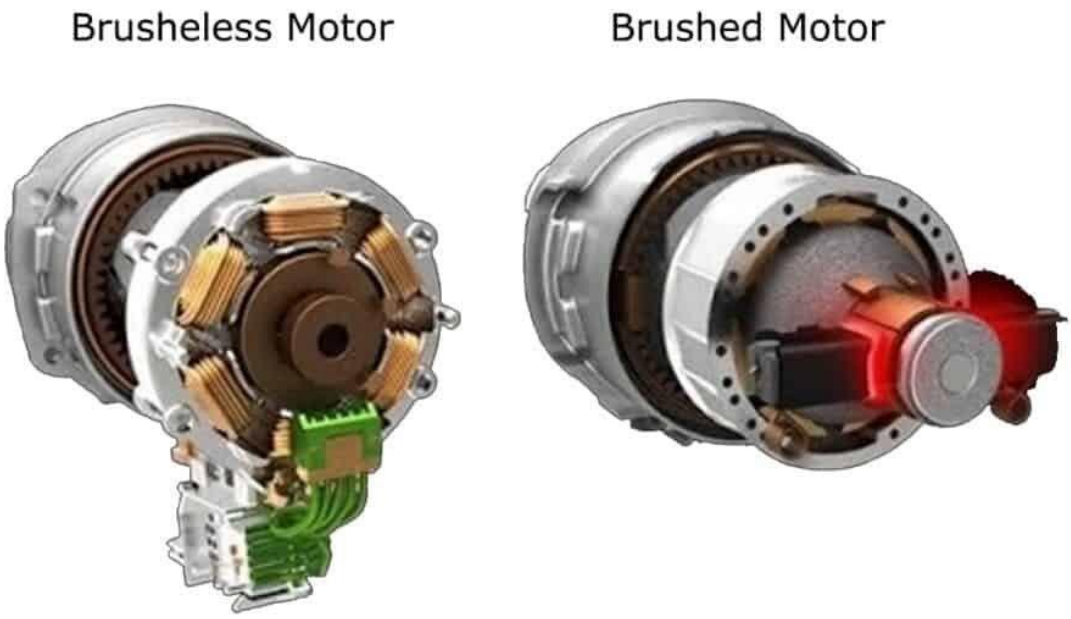


Figure 2.2.2.1

BLDC controller(Brushless DC motor controller)

Most quality e-bikes are mounted with brushless DC motors that feature permanent magnets. They are more efficient, have better power-to-weight ratios, and are quite reliable than the brushed motors that produce overheating.

The controller corresponding to the brushless motor is the BLDC which usually has 3 phases controlled by a set of keys.

Brushed DC motor controllers

Another type of motor used in electric bikes is the brushed DC motor. The brushed controllers have a much simpler design to regulate this type of motors that are often used in budget e-bike options.

BLDC controllers for motors with Hall Sensors

There are permanent magnets on the rotor, and electromagnets are on the stator. The hall sensors are on the controller's circuit board that monitors and controls the rotation. They are used to manage the timing of the power delivery when we need to push the rotor.

Figure 2.2.2.2

2.2.3: How to choose the right controller

If you want to use a controller with your electric bike, you need to take several factors into account.

To match the voltage

The first thing is to ensure that the controller can handle the voltage and current requirements. That means you need to check whether the controller voltage and power are the same as those of the motor when selecting because overheating is a likely consequence of running too much voltage through a controller.

With the correct voltage and power, you'll reduce the heat generated by the controller and increase the stability and reliability of its operation.

To match the power rating

Motors will not be able to reach its maximum speed as its claimed if an unmatched controller controls it.

Running an unmatched controller either higher or lower volts than the motor and battery will not only affect the motor's performance but also increase the risk of overheating and connectors burning out.

The motor controller you select should be compatible with the motor's power rating and the battery voltage you are using.

To match the type of the motor

You should also consider the controller type to match the current motor type used on your e-bike.

Brushless e-bike controller

You will need to choose the brushless motor controller if your ebike is built with a brushless motor.

Sensored controller

Some controllers require hall sensors, while others can run without them. Hall sensors on the motor detect its rotation, and the controller produces power according to the sensor signals. So you should choose the controller mode with hall sensor if your E-bike motor is sensor enabled.

Using a brushless motor controller that supports a sensorless operation is necessary if you've got a sensorless motor.

Sine wave controller

Among the most popular types of controllers is the sine wave controller, which is well known for its low noise output.

It helps your bike in better performance when climbing hills or carrying loads. On the other hand, the sine wave controllers will use much more power and, therefore, will cost you very much more. Furthermore, they are limited to matched motors
You also need to know that a sine wave controller can provide smoother acceleration with switched reluctance motors.

Square wave controller

The square wave is another type of controller that differs in the phase voltage waveform of the sine wave controller. It generates a rectangular waveform.
In comparison to sine wave controllers, square wave controllers are less expensive and are more robust during sudden braking or acceleration. They can also work with different motor types
However, there are some disadvantages of square wave controllers: they usually generate high noise and cannot produce even or smooth control, also, square wave controllers have difficulty increasing motor efficiency when going uphill or carrying a heavy load

2.2.4: Circuit Diagram

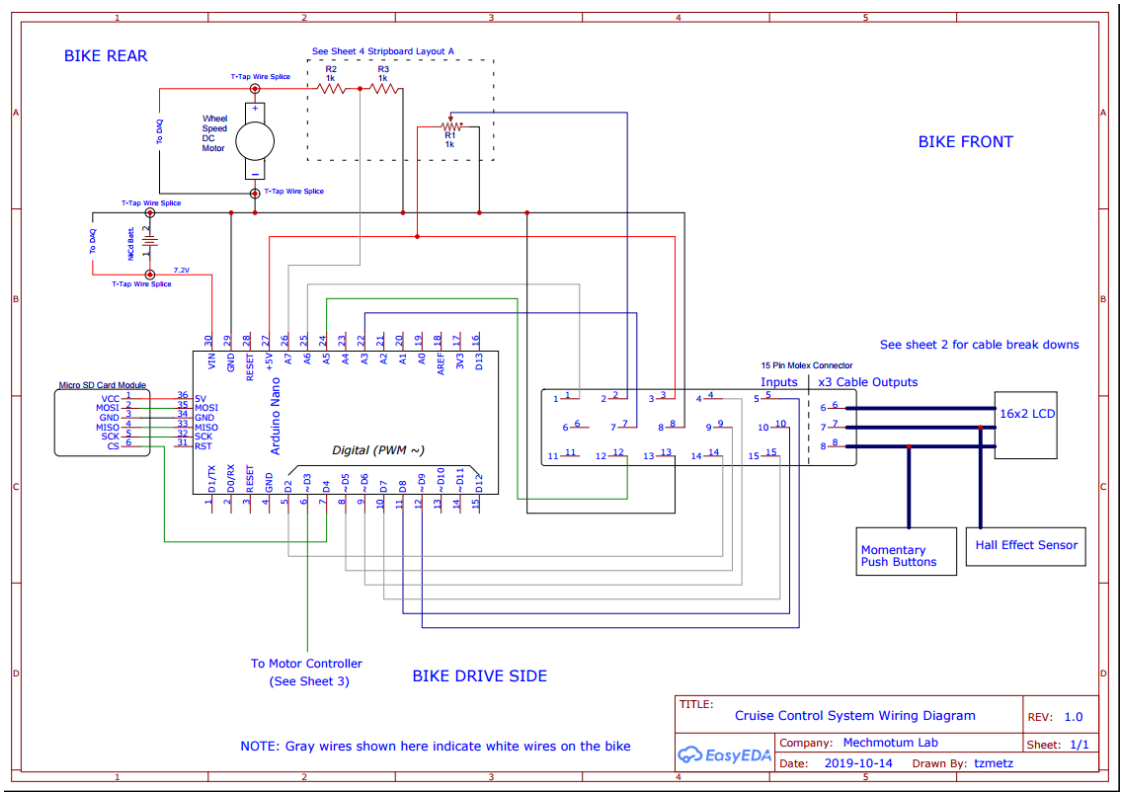


Figure 2.2.4.1

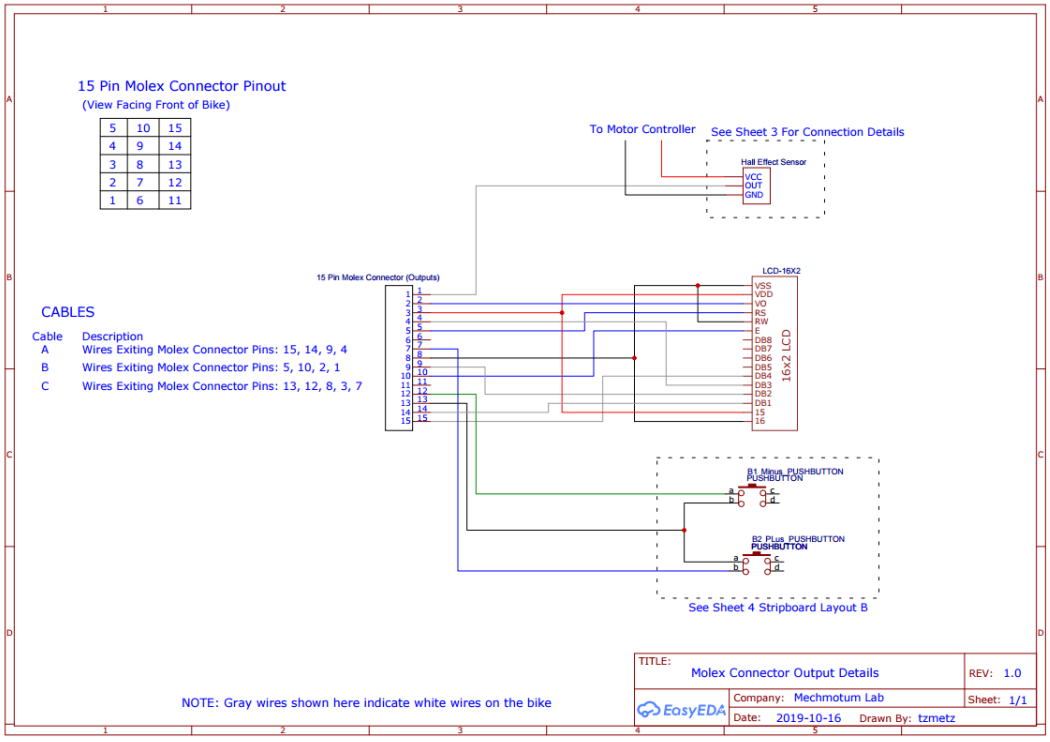


Figure 2.2.4.2

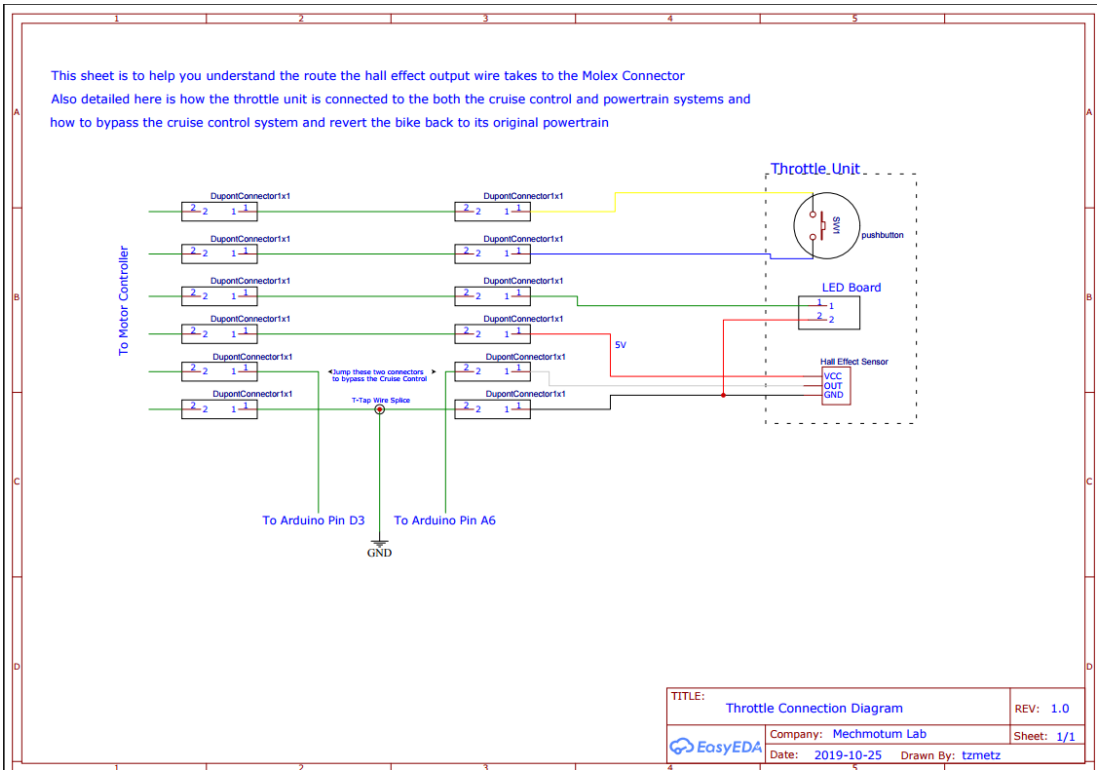


Figure 2.2.4.3

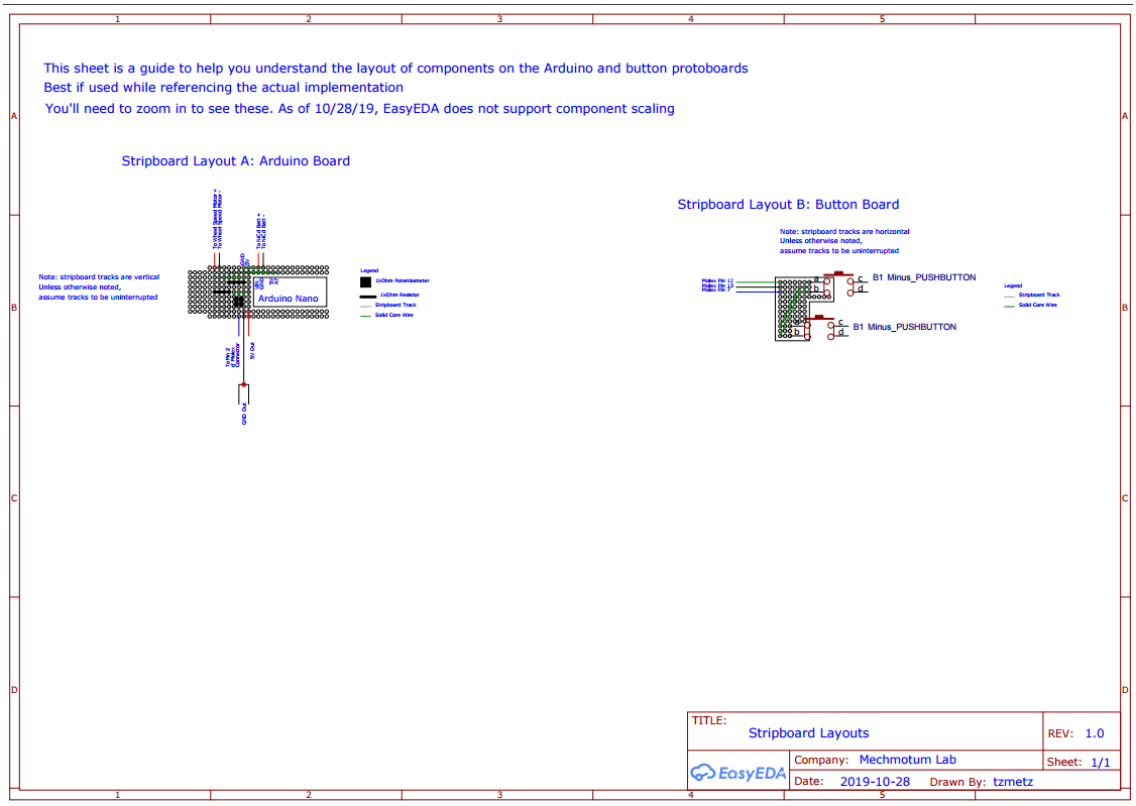


Figure 2.2.4.4

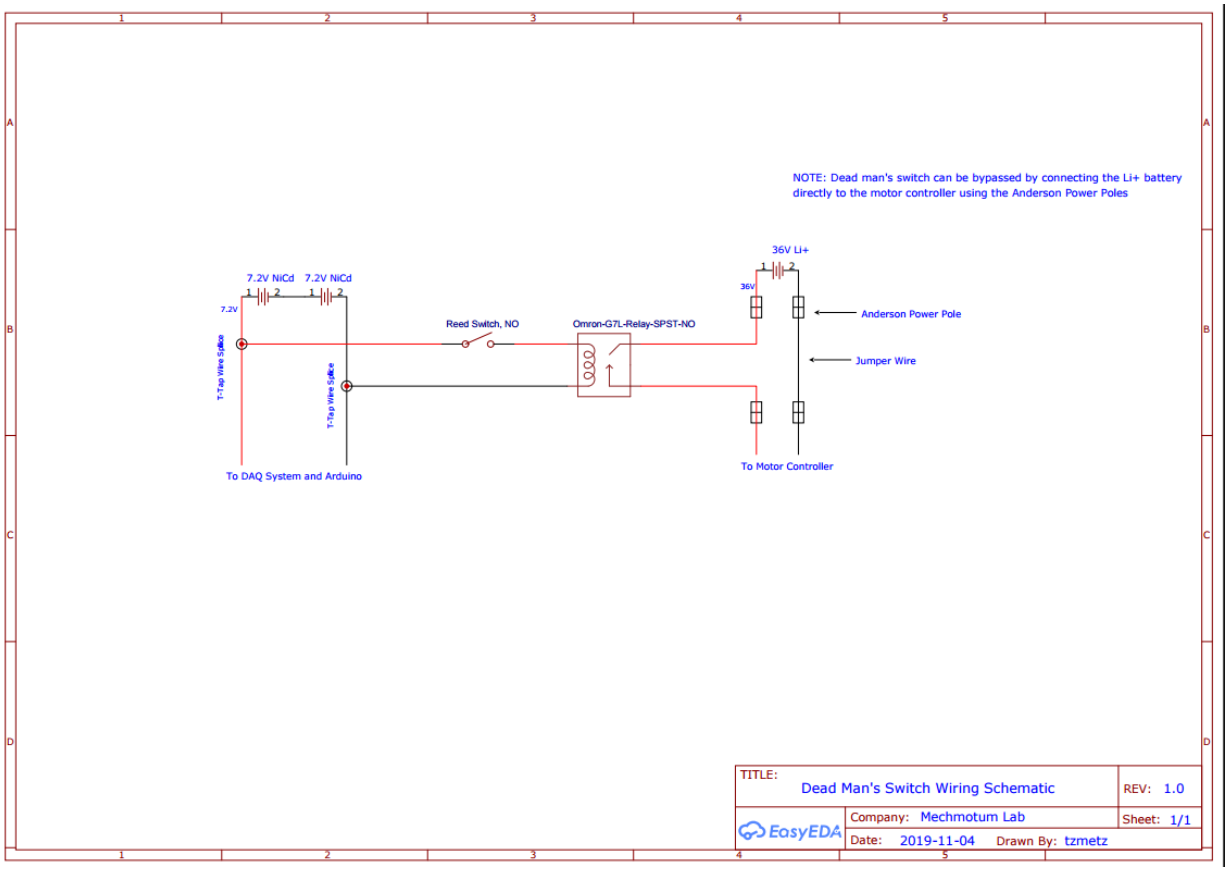


Figure 2.2.4.5

2.2.5: Hardware Implementation

Figure 2.2.5.1

Controller Pinouts

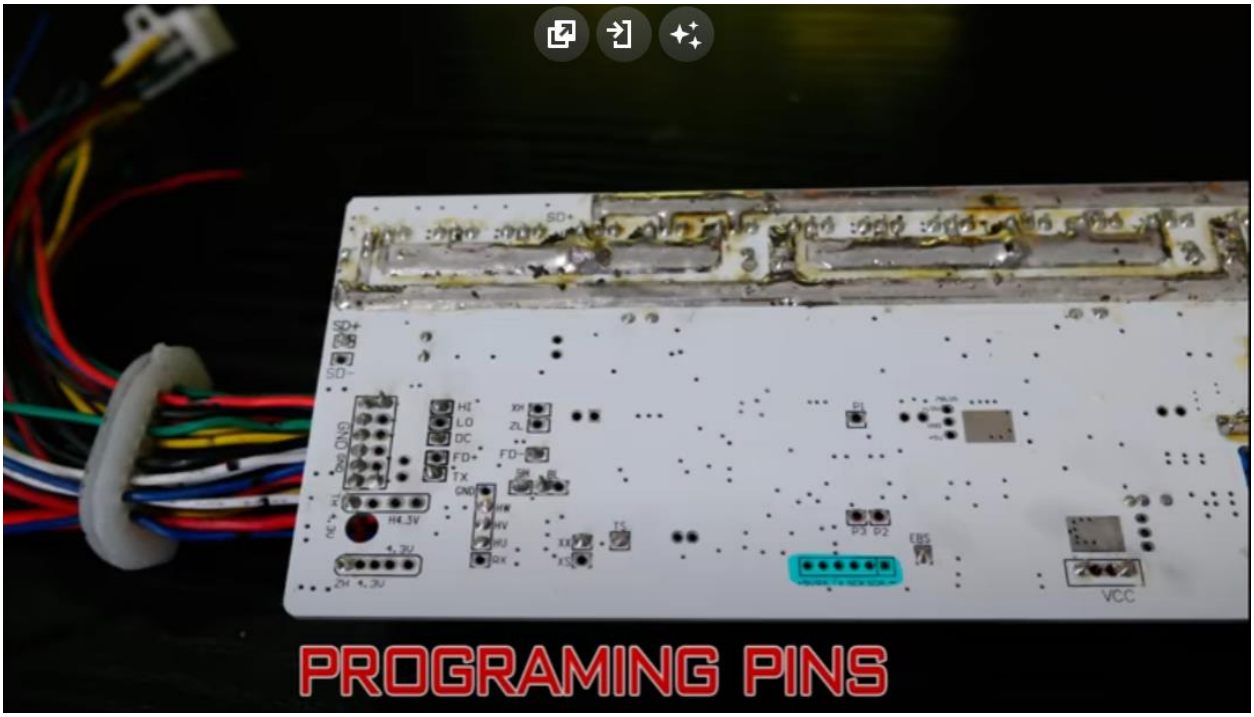


Figure 2.2.5.2

Throttle Activation

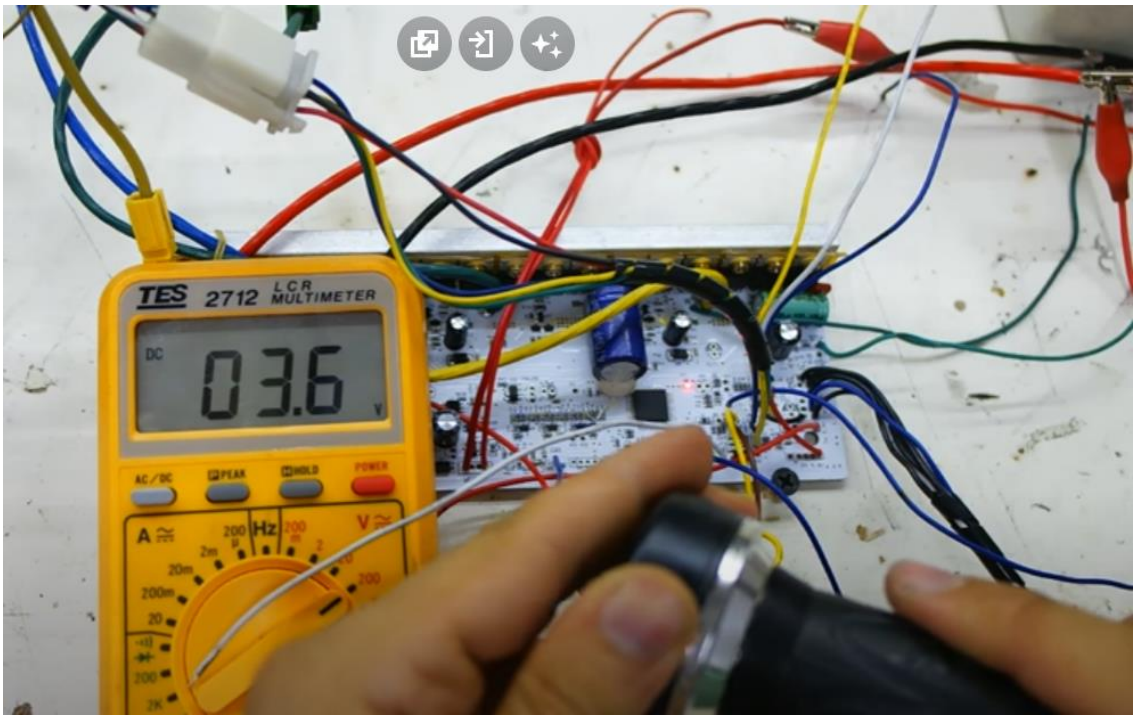


Figure 2.2.5.3

2.3: Battery:

2.3.1: Functionalities

What is a lithium-ion battery and how does it work?

A lithium-ion (Li-ion) battery is an advanced battery technology that uses lithium ions as a key component of its electrochemistry. During a discharge cycle, lithium atoms in the anode are ionized and separated from their electrons. The lithium ions move from the anode and pass through the electrolyte until they reach the cathode, where they recombine with their electrons and electrically neutralize. The lithium ions are small enough to be able to move through a micro-permeable separator between the anode and cathode. In part because of lithium's small size (third only to hydrogen and helium), Li-ion batteries are capable of having a very high voltage and charge storage per unit mass and unit volume.

Li-ion batteries can use a number of different materials as electrodes. The most common combination is that of lithium cobalt oxide (cathode) and graphite (anode), which is most commonly found in portable electronic devices such as cellphones and laptops. Other cathode materials include lithium manganese oxide (used in hybrid electric and electric automobiles) and lithium iron phosphate. Li-ion batteries typically use ether (a class of organic compounds) as an electrolyte.

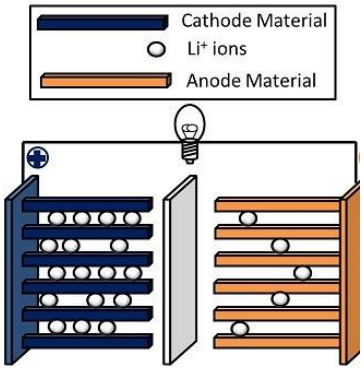


Figure 2.3.1.1

What are some advantages of Li-ion batteries?

Compared to the other high-quality rechargeable battery technologies (nickel-cadmium or nickel-metal-hydride), Li-ion batteries have a number of advantages. They have one of the highest energy densities of any battery technology today (100-265 Wh/kg or 250-670 Wh/L). In addition, Li-ion battery cells can deliver up to 3.6 Volts, 3 times higher than technologies such as Ni-Cd or Ni-MH. This means that they can deliver large amounts of current for high-power applications, which has Li-ion batteries are also comparatively low maintenance, and do not require scheduled cycling to maintain their battery life. Li-ion batteries have no memory effect, a detrimental process where repeated partial discharge/charge cycles can cause a battery to 'remember' a lower capacity. This is an advantage over both Ni-Cd and Ni-MH, which display this effect. Li-ion batteries also have low self-discharge rate of around 1.5-2% per month. They do not contain toxic cadmium, which makes them easier to dispose of than Ni-Cd batteries.

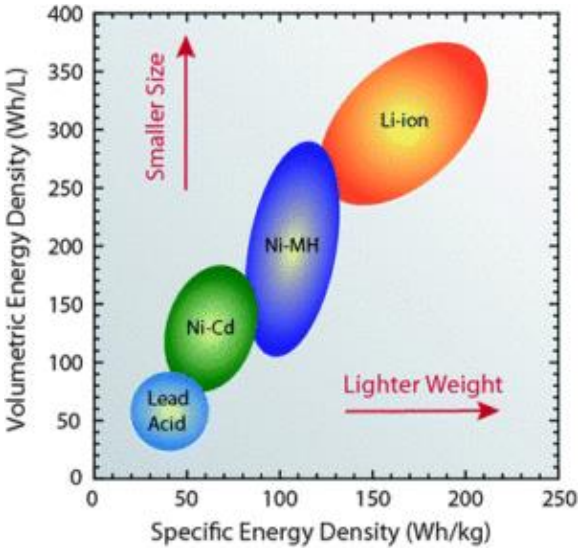


Figure 2.3.1.2

Due to these advantages, Li-ion batteries have displaced Ni-Cd batteries as the market leader in portable electronic devices (such as smartphones and laptops). Li-ion batteries are also used to power electrical systems for some aerospace applications, notable in the new and more environmentally friendly Boeing 787, where weight is a significant cost factor. From a clean energy perspective, much of the promise of Li-ion technology comes from their potential applications in battery-powered cars. Currently, the bestselling electric cars, the Nissan Leaf and the Tesla Model S, both use Li-ion batteries as their primary fuel source.

What are some disadvantages of Li-ion batteries?

Despite their technological promise, Li-ion batteries still have a number of shortcomings, particularly with regards to safety. Li-ion batteries have a tendency to overheat, and can be damaged at high voltages. In some cases, this can lead to thermal runaway and combustion. This has caused significant problems, notably the grounding of the Boeing 787 fleet after onboard battery fires were reported. Because of the risks associated with these batteries, a number of shipping companies refuse to perform bulk shipments of batteries by plane. Li-ion batteries require safety mechanisms to limit voltage and internal pressures, which can increase weight and limit performance in some cases. Li-ion batteries are also subject to aging, meaning that they can lose capacity and frequently fail after a number of years. Another factor limiting their widespread adoption is their cost, which is around 40% higher than Ni-Cd. Addressing these issues is a key component for current research into the technology. Finally, despite the high energy density of Li-ion compared to other kinds of batteries, they are still around a hundred times less energy dense than gasoline (which contains 12,700 Wh/kg by mass or 8760 Wh/L by volume).

2.3.2: Connection

Series and parallel batteries' connections

Series connection (also called Series circuit) increases the operating voltage. If a higher capacity of electricity storage is required, a parallel connection of lithium battery bank is necessary. In some energy storage applications, combinations of series and parallel connection are also used.

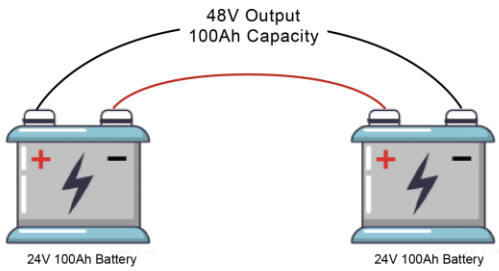


Figure 2.3.2.1

Example: If two batteries of 200Ah (amp-hours) and 24V (volts) each are connected in series, the resulting output voltage is 48V with a capacity of 200 Ah.

Lithium Batteries are Connected in Series as Follows

Any number of lithium batteries are usually connected in series. The negative pole of one lithium battery is connected to the positive pole of the other lithium battery so that the same current flows through all batteries. The resulting total voltage is then the sum of the partial voltages. Series circuit of lithium batteries is often referred to as series connection.

Advantages and Disadvantages of the Lithium Battery Series Connection

Series connection of lithium batteries makes it possible to produce higher total voltages in this way. However, the disadvantage is that the weakest lithium-ion

battery cell affects the performance of the entire series. In the worst case, a defective lithium solar battery can cause the entire lithium battery storage series to fail. Therefore, additional fuses must be added in series.

In the case of compact solar power storage systems, it also happens that the entire must be replaced if a lithium-ion battery cell is broken. The control (cell balancing) of a lithium-ion battery backup based on a series connection also leads to performance losses. As a rule, only battery cells of the same manufacturer, type and lithium-ion battery technology can be connected in series.

This is How Lithium Batteries are Connected Together in Parallel

When lithium batteries are connected in parallel, the positive terminal is connected to the positive terminal and the negative terminal is connected to the negative terminal. The charge capacity (Ah) of the individual lithium-ion batteries then adds up while the total voltage is equal to the voltage of the individual lithium ion batteries. As a general rule, only lithium-ion batteries of the same voltage and acid density with the same state of charge should be connected together in parallel, and wire cross-sections and lengths should also be exactly the same.

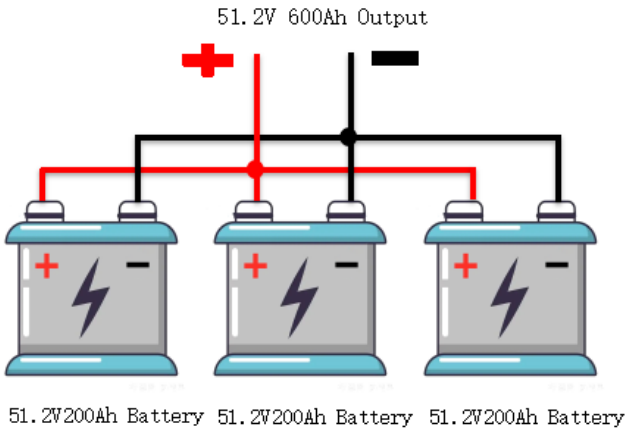


Figure 2.3.2.2

Example: If two batteries, each with 200 Ah and 51.2V, are connected in parallel, this results in an output voltage of 51.2V and a total capacity of 600Ah.

Advantages and Disadvantages of the Lithium Battery Parallel Connection

In this way, with a parallel connection of lithium battery bank, the performance and also the service life can be increased. Used as power storage for PV power, parallel connection of lithium solar batteries can reduce the cost of home battery storage system and create new opportunities in energy storage construction, unlike series connection.

Nevertheless, the charge control of lithium solar batteries connected in parallel is not straightforward, as each battery is also subject to aging and thus a potential source of error. Therefore, it can also be advantageous to plan for one large lithium battery bank instead of several small lithium ion batteries.

If a parallel connection is nevertheless made, care should be taken, as with series connection, to use lithium batteries of the same type, capacity, age and state of charge. In addition, the shortest possible cable runs should be used to prevent major voltage losses.

However, newer cell electronics placed on each lithium-ion battery cell make it possible to combine lithium batteries regardless of size, manufacturer, storage technology, and performance in a power storage system by continuously measuring the capacity during operation. In this way, lithium solar batteries connected in parallel can be gradually upgraded during operation.

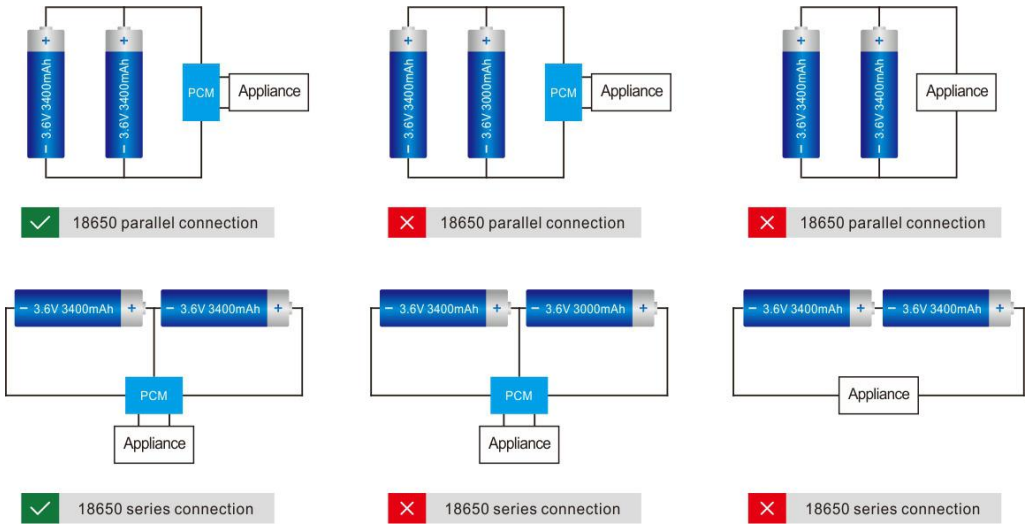


Figure 2.3.2.3

2.3.3: Implementation & Installation

How to Calculate a Lithium-Ion Battery Pack's Capacity and Runtime

Capacity Varies With Load Current - Batteries have a nominal capacity, but their real capacity depends on the current being drawn from them.

Capacity is a function of the type of battery you are using, the load current, temperature and age of the cell. The capacity of lithium-ion batteries can be reduced by as much as 25% at high current (C rating) and operating temperature as compared to their published capacity. Manufacturers typically publish the the capacity when the load is C/5 or one fifth of the rated capacity. When you are trying to determine the capacity at your use conditions, a bit more engineering is required.

Batteries have internal impedance which reduces capacity and runtime at high load currents. Basic battery calculators don't account for the reduction in capacity. The best solution is to generate empirical cycling data at the desired current or use an advanced battery calculator that accounts for the cell's unique impedance profile.

What is a Cell's Rated Capacity?

A Lithium Ion battery's published rated capacity is the capacity of the cell when the load current is one fifth of the rated capacity (the C Rate). When the current varies from C/5, the capacity will change due to chemical reaction rates including a chemical effect called concentration polarization.

What Is the Capacity of a Cell When the Load Current Increases?

If the load current is greater than one fifth of rated capacity, the capacity and runtime decreases. The cell's capacity will be less than the rated capacity.

What Is the Capacity of a Cell When the Load Current Decreases?

If the load current is lower than the rated capacity, the capacity and runtime increases.The cell's capacity will be grater than the rated capacity.

Implementation Steps

1.Calculations

First of all we need to do some maths to calculate how many batteries we need, how many batteries will be connected in series and in parallel .

We need a battery with a 48 V and 12 Ah capacity so by dividing The total capacity we need on the capacity of one single battery which is 2000 mAh that gives us the number of cells will be connected in parallel which is six io our case. Then we will divide the total volts we need on the vault of each battery which is 3.7 that's give us about 13 battery to be connected in series. Then by multiplying the number of series and parallel connected batteries we have the number of total cells we need which is 78.

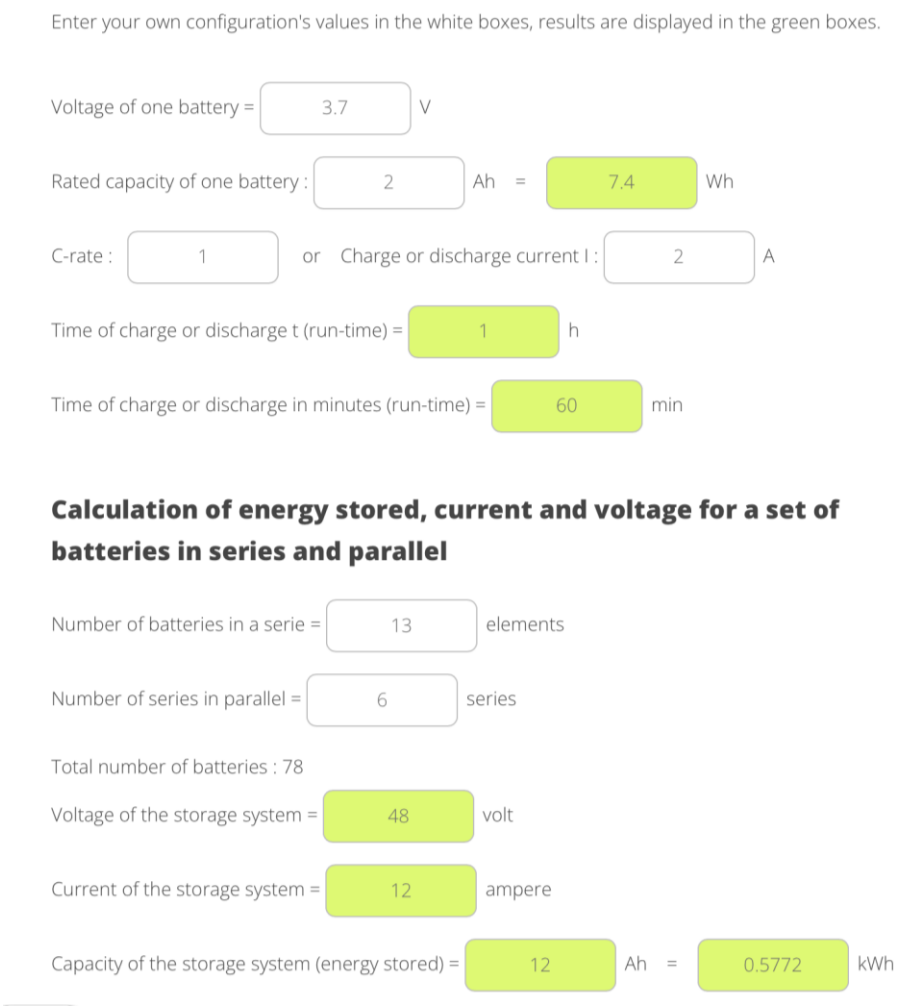


Figure 2.3.3.1

2.Check voltage

Next we need to check each battery to be almost 3.7 V To make sure that they are connected safely and correctly. This step is important because if there is a cell is not working well the whole battery will be affected and may not work well.

3.Draw a sketch

The third step is optional but it gives us more help to imagine how the battery will be shaped .we can draw a sketch to show the connections series and parallel of each cell in the battery and The final shape of the battery.

4. Glow the cells

Step four is to glue cells together as the shape which shows on the previous step. This step prepare us to use the nickel wire we have and the spot welding machine to make the connections between the cells themselves.

5. Series and parallel connections

In this step we will use the nickel wire and the spot welding to make the series and parallel connection as in the sketch we drawn. After finishing the connections on one side of the battery will Flip it and re-do the connections on the other side.

6.BMS Connection

And this time we will glue the BMS to the battery and start to set it up by connecting the negative wire to the negative side of the battery as the positive to the positive side of the battery after that we will take the small wires and start connecting them to the battery on each side positive and negative. Battery management system is important to make sure the battery is safe and is rechargeable and also to make reports about the condition of the battery.

7. Cover up the battery

The last step is to cover the battery and heat the cover up to make sure the cover is settled perfectly on the battery. Be careful not to use too much heat or you will damage the battery.

Check battery health

It is important to make sure that our battery is working well and has no troubles. We can do that by making a regular battery check using an Avometer to check the battery's condition.

In the upcoming table are 48 V battery readings according to its condition:

48V LiFePO4 Battery Charging Parameters

- **Charging voltage:** 56.8–58.4V
- **Float voltage:** 54.4V (or disabled)
- **Maximum voltage:** 58.4V
- **Minimum voltage:** 40V
- **Nominal voltage:** 48V or 51.2V

Figure 2.3.3.2

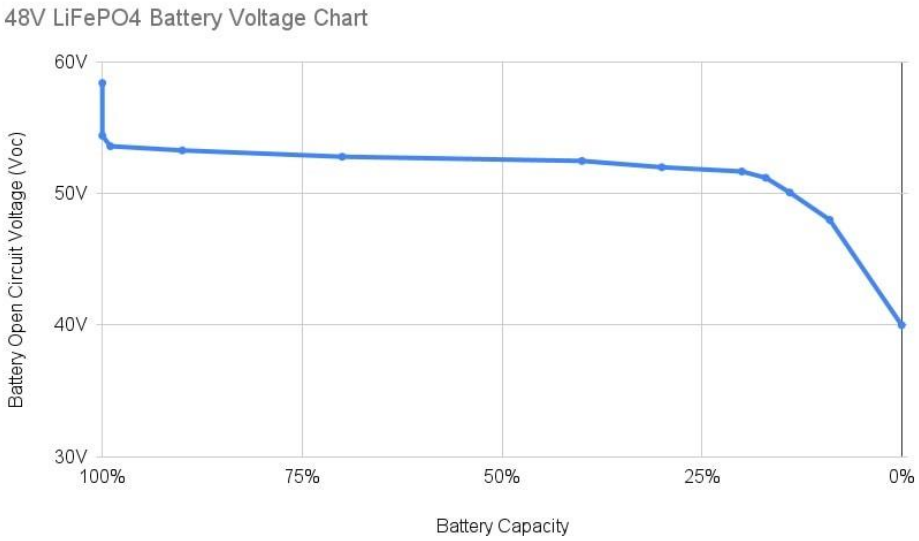


Figure 2.3.3.3

48V LiFePO4 Battery Voltage Chart

Voltage	Capacity
58.4V	100% (charging)
54.4V	100% (resting)
53.6V	99%
53.2V	90%
52.8V	70%
52.4V	40%
52.0V	30%
51.6V	20%
51.2V	17%
50.0V	14%
48.0V	9%
40.0V	0%

Figure 2.3.3.4

2.4: Peripherals:

2.4.1: Throttle

An e-bike throttle is a control mechanism that allows the rider to control the speed and power of the electric motor on their electric bicycle without pedaling. It is typically located on the handlebars of the bike and can be activated by the rider's hand.

There are several types of e-bike throttles available, including twist grips, thumb throttles, and trigger throttles. Twist grips are the most common type and require the rider to twist the grip on the handlebar to increase or decrease the power of the electric motor.

Thumb throttles are another popular type of e-bike throttle that is operated by pressing a small lever with the rider's thumb. This type of throttle is often more intuitive to use than a twist grip and can be more comfortable for riders with smaller hands.

Trigger throttles are less common but can provide a more precise control experience. They are operated by pulling or pushing a trigger located on the handlebar, which can allow the rider to make finer adjustments to the motor's power output.

E-bike throttles can be a useful feature for riders who want to conserve their energy or have limited mobility. They can also be helpful for commuting or riding in hilly areas where additional power may be required to climb steep inclines.

It is important to note that some countries and regions have regulations regarding the use of e-bike throttles, including speed limits and restrictions on where e-bikes can be ridden. It is important to check local laws and regulations before using an e-bike throttle.

2.4.2: Pedal Assist Sensor (PAS)

A Pedal Assist Sensor (PAS) is a sensor system that is commonly used on electric bicycles to detect the rider's pedaling motion and provide assistance from the electric motor. The PAS is typically located near the bottom bracket of the bike and is connected to the motor and the bike's control system.

The PAS works by detecting the movement of the pedals and sending a signal to the motor to provide assistance. The amount of assistance provided is determined by the level of pedal input from the rider, with more assistance provided as the rider pedals harder.

PAS systems can be designed to provide different levels of assistance, ranging from low to high, depending on the rider's needs. Some systems may also allow the rider to adjust the level of assistance through a control panel or display on the handlebars.

One of the benefits of a PAS system is that it can provide a more natural and intuitive riding experience for the rider. The assistance from the motor is directly tied to the rider's pedaling motion, which can make it easier to control the bike and maintain a consistent speed.

PAS systems can also be more efficient than throttle-based systems, as the assistance is only provided when the rider is pedaling. This can help to extend the range of the e-bike and conserve battery power.

2.4.3: E-Brakes

E-brakes, also known as electric brakes, are a type of braking system that is commonly used on electric bicycles (e-bikes). E-brakes work by using an electric signal to activate the brake calipers, which then apply pressure to the wheel rims or disc rotors to slow or stop the bike.

There are two main types of e-brakes: regenerative and non-regenerative. Regenerative e-brakes use the motor's power to generate electricity when the brakes are applied, which is then stored in the battery for later use. This can help to extend the e-bike's range by reducing the amount of energy that is lost during braking.

Non-regenerative e-brakes, on the other hand, do not generate electricity when the brakes are applied. Instead, they rely on a direct electric signal to activate the brake calipers and slow or stop the bike.

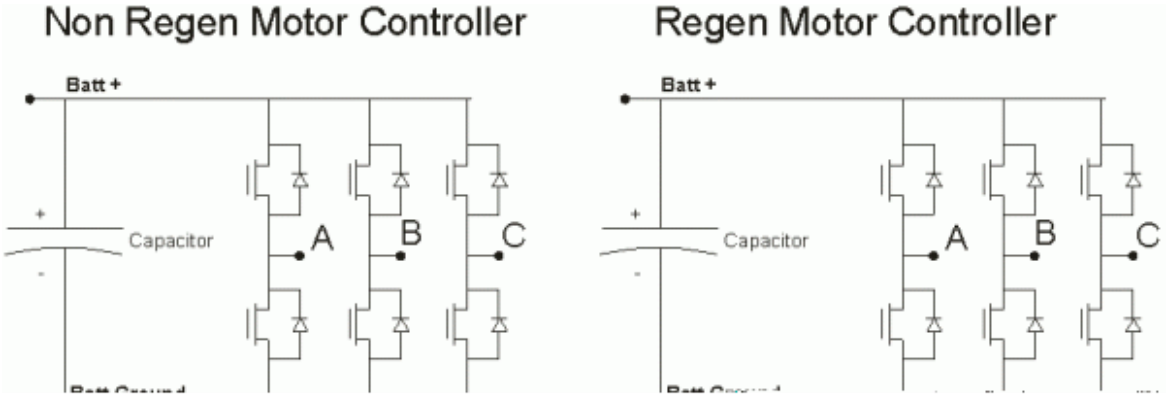


Figure 2.4.3.1

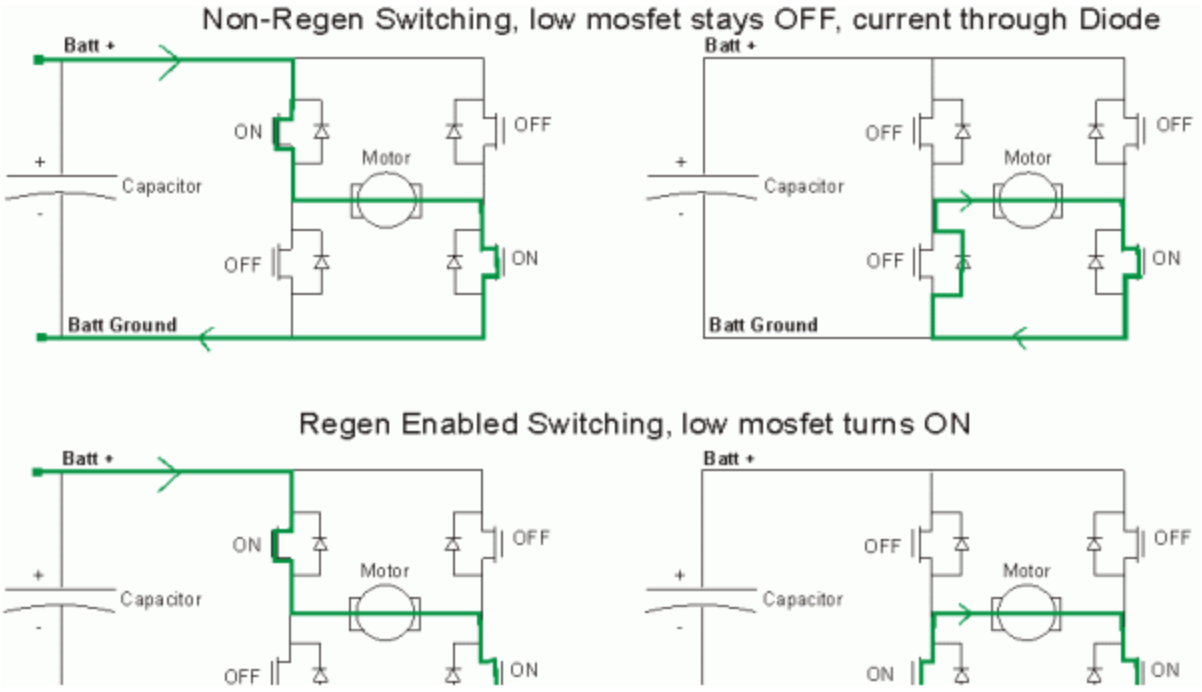


Figure 2.4.3.2

E-brakes can provide several benefits for e-bike riders. One of the main benefits is that they can provide more reliable and consistent braking performance than traditional mechanical brakes, especially at high speeds or in wet or slippery conditions. They can also be more efficient, as the electric signal can provide faster and more precise control over the braking force.

Another benefit of e-brakes is that they can help to reduce wear and tear on the brake pads and rims or rotors, which can extend the lifespan of these components and reduce maintenance costs over time.

2.4.4: LCD Screen

An LCD screen, or Liquid Crystal Display screen, is a type of display that is commonly used on electric bicycles (e-bikes) to provide information about the bike's performance and settings. The LCD screen is typically located on the handlebar or stem of the bike and is connected to the bike's control system.

The LCD screen can display a variety of information, including the bike's speed, distance traveled, battery level, and assist level. Some screens may also display information about the motor's power output, cadence, and other performance metrics.

One of the benefits of an LCD screen is that it can provide the rider with real-time feedback about their performance and the e-bike's performance. This can help the rider to make adjustments to their riding style or assist level to optimize their efficiency and conserve battery power.

LCD screens can also be customizable, with some systems allowing the rider to adjust the screen's brightness, font size, and other settings to suit their preferences. Some screens may also incorporate additional features, such as GPS navigation or Bluetooth connectivity.

2.5: Finish & Cable Management:

Motor Installed

Figure 2.5.1

Controller & Battery & Peripherals Installed

Figure 2.5.2

IoT Sensor Network Installed

Figure 2.5.3

Finished Prototype

Figure 2.5.4

Figure 2.5.5

Figure 2.5.6

Chapter 3: Secondary Components (Sensor Network)

3.1: NodeMCU (Wi-Fi Module ESP32)

3.1.1: Functionality

IoT (Internet of Things) is a technology that allows different devices and objects to connect to the internet, share data, and communicate with each other. This technology includes everything from household appliances to cars, buildings, and entire cities. IoT is used in many different applications, such as industry, healthcare, agriculture, transportation, and smart cities.

IoT is used in many different applications, such as:

Industry: where IoT is used to improve productivity, reduce costs, and increase safety.

Health: where IoT is used to monitor health and fitness and improve healthcare.

Agriculture: where IoT is used to improve crop productivity, reduce water and energy consumption.

Transportation: where IoT is used to improve safety and efficiency in transportation and reduce maintenance costs.

Smart cities: where IoT is used to improve resource management, save energy and water, and improve public services.

challenges and risks: -

Although IoT offers many benefits, it also faces several challenges and risks. Among the key challenges facing IoT are:

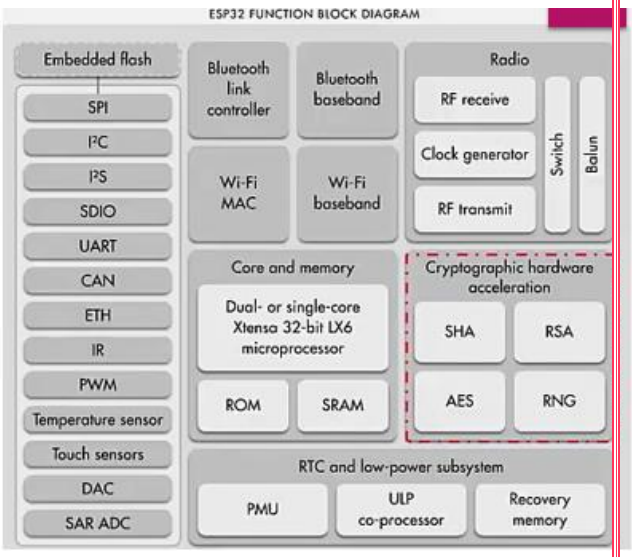
Security: Security and protection against cyber-attacks and breaches are one of the biggest challenges facing IoT. For example, hackers can target IoT devices and use them to access personal or sensitive data.

Interoperability: Interoperability between different IoT devices is also a challenge, as devices from different manufacturers may not be compatible with each other. This can hinder the development of integrated IoT solutions and limit the ability of businesses and organizations to leverage the full potential of IoT.

Data Privacy: IoT devices collect and transmit a vast amount of data, including personal and sensitive information. Therefore, it is essential to ensure that strict data privacy measures are in place to protect users' privacy and comply with data protection regulations.

Complexity: Developing and implementing IoT systems can be complex, requiring specialized skills and expertise. This can be a challenge for businesses or organizations that lack the necessary resources or expertise.

The data collected by IoT devices can be analyzed using data analytics tools and machine learning algorithms to extract valuable insights and patterns. For example, in agriculture, IoT sensors can monitor soil moisture levels, temperature, and humidity, allowing farmers to optimize irrigation and fertilization and increase crop yields. In healthcare, IoT devices can monitor patients' vital signs and alert healthcare providers of any abnormal readings, enabling early intervention and faster response times.



However, as with any technology, IoT also poses significant challenges and risks. One of the main challenges is security, as IoT devices are vulnerable to cyber-attacks and data breaches. This is particularly concerning in industries such as healthcare, where the data collected by IoT devices can be sensitive and personal. Another challenge is the interoperability of different IoT devices, as devices from different manufacturers may not be compatible with each other.

To address these challenges, it is crucial to develop secure and interoperable IoT devices and platforms, implement strict security measures, and establish industry standards and regulations. Collaboration between government agencies, industry stakeholders, and technology providers is also crucial to ensure the successful development and implementation of IoT solutions.

In conclusion, IoT presents a vast potential to transform various industries and sectors by enabling smart applications and solutions that can improve efficiency, reduce costs, and enhance safety. However, it also poses significant challenges that need to be addressed to ensure its successful implementation and minimize the risks and challenges it presents. By developing secure and interoperable IoT devices and platforms, implementing strict security measures, and establishing industry standards and regulations, we can realize the full potential of IoT while ensuring the privacy and security of users' data. Figure 3.1.1.1

ESP32

Processors:

Main processor: Tensilica Xtensa 32-bit LX6 microprocessor

Cores: 2 or 1 (depending on variation)

Clock frequency: up to 240 MHz

Performance: up to 600 DMIPS

Ultra low power co-processor: allows you to do ADC conversions, computation, and level thresholds while in deep sleep.

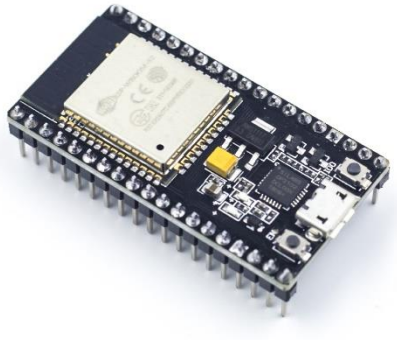


Figure 3.1.1.2

Wireless connectivity:

Wi-Fi: 802.11 b/g/n/e/i (802.11n @ 2.4 GHz up to 150 Mbit/s)

Bluetooth: v4.2 BR/EDR and Bluetooth Low Energy (BLE)

Memory:

Internal memory:

ROM: 448 KB

For booting and core functions.

SRAM: 520 KB

For data and instruction.

RTC fast SRAM: 8 KB

For data storage and main CPU during RTC Boot from the deep-sleep mode.

RTC slow SRAM: 8 KB

For co-processor accessing during deep-sleep mode.

eFuse: 1 KB

Of which 256 bits are used for the system (MAC address and chip configuration) and the remaining 768 bits are reserved for customer applications, including Flash-Encryption and Chip-ID.

Embedded flash:

External flash & SRAM: ESP32 supports up to four 16 MiB external QSPI flashes and SRAMs with hardware encryption based on AES to protect developers' programs and data. ESP32 can access the external QSPI flash and SRAM through high-speed caches.

- Up to 16 MiB of external flash are memory-mapped onto the CPU code space, supporting 8-bit, 16-bit and 32-bit access. Code execution is supported.
- Up to 8 MiB of external flash/SRAM memory are mapped onto the CPU data space, supporting 8-bit, 16-bit and 32-bit access. Data-read is supported on the flash and SRAM. Data-write is supported on the SRAM.

ESP32 chips with embedded flash do not support the address mapping between external flash and peripherals.

Peripheral input/output: Rich peripheral interface with DMA that includes capacitive touch, ADCs (analog-to-digital converter), DACs (digital-to-analog converter), I²C (Inter-Integrated Circuit), UART (universal asynchronous receiver/transmitter), CAN 2.0 (Controller Area Network), SPI (Serial Peripheral Interface), I²S (Integrated Inter-IC Sound), RMII (Reduced Media-Independent Interface), PWM (pulse width modulation), and more.

Security:

- IEEE 802.11 standard security features all supported, including WFA, WPA/WPA2 and WAPI
- Secure boot
- Flash encryption
- 1024-bit OTP, up to 768-bit for customers
- Cryptographic hardware acceleration: AES, SHA-2, RSA, elliptic curve cryptography (ECC), random number generator (RNG)

3.1.2: Pinouts

PINS: -

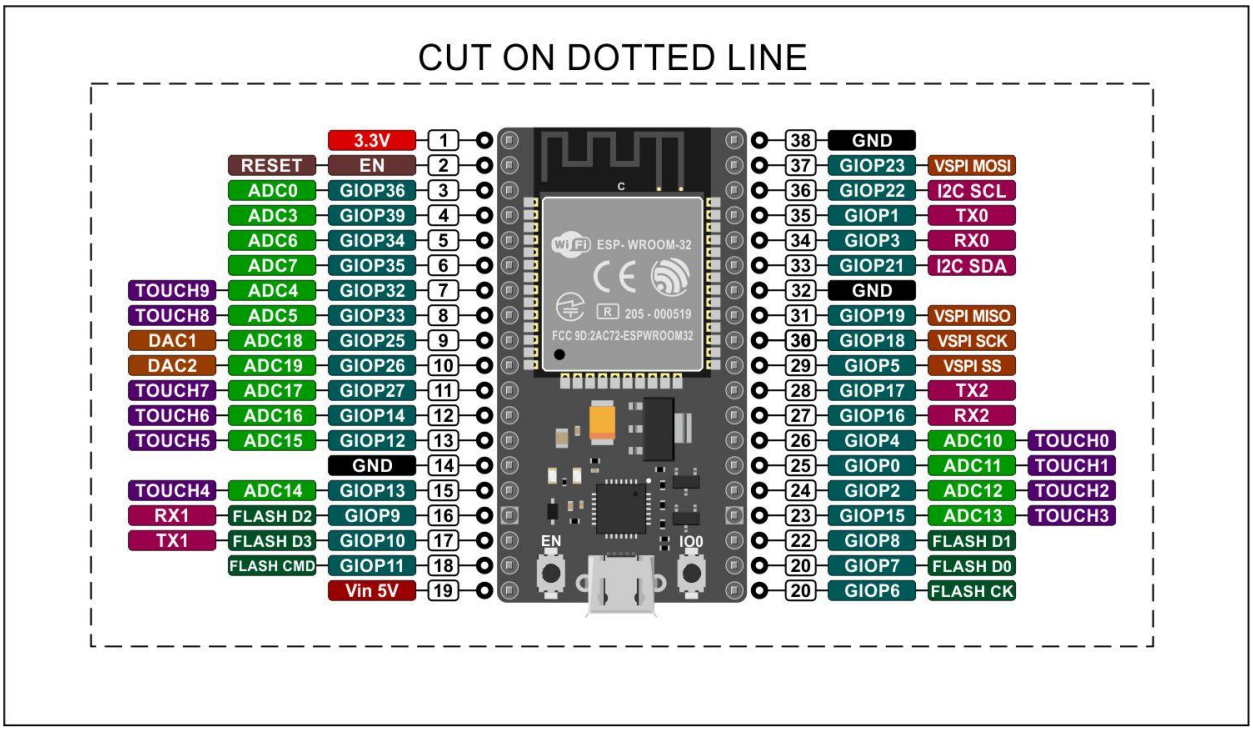


Figure 3.1.2.1

The ESP32 consists of 38 controllable pins and has many features that make it suitable for a wide range of applications, including smart device control, IoT, robotics, medical devices, and more.

1. GPIO pins (34 pins): These are the pins that can be used to control other electronic devices. They are distributed on both sides of the module, with 18 pins on the right side and 16 pins on the left side. They can be used as inputs or outputs and support many functions such as PWM, ADC, SPI, I2C, and UART.

2. Power and Ground pins (2 pins): These pins are used to connect the ESP32 to a power source and ground. The ESP32 has two power pins: Vin and VUSB. Vin can be connected to a power source ranging from 5V to 12V, while VUSB is used to connect the ESP32 to a USB power source.

3. Communication pins (2 pins): This set of pins allows you to control the ESP32's wireless communication module, including Wi-Fi, Bluetooth, and BLE. The ESP32 has two communication pins: TX and RX.

4. Memory pins (3 pins): These pins are used to control various memory modules, such as SD and SPI Flash. The ESP32 has three memory pins: SD_CLK, SD_DATA_0, and SD_DATA_1.

5. EN pin (1 pin): This pin is used to enable and disable the ESP32. When the pin is connected to ground, the ESP32 is turned off.

6. BOOT pin (1 pin): This pin is used to load the initial bootloader program in the ESP32.
7. LED lighting pins (2 pins): The ESP32 has two pins for LED lighting, which are used to show the operating and transmission status of the module.
In addition to the above-mentioned pin groups, the ESP32 has other pins for accessing other module components such as speakers, sensors, and power systems.
The ESP32 is flexible, adaptable, and supports a wide range of protocols such as Wi-Fi, Bluetooth, BLE, MQTT, HTTPS, and more. It can be used to control many different devices such as motors, lights, sensors, and screens.
Thanks to the controllable pins feature, users can customize the ESP32 to their specific needs and easily connect it to other devices. The ESP32 also supports low-power applications that require low power consumption, making it suitable for many applications in the IoT field.

PROTOCOLS: -

The ESP32 is a powerful and versatile microcontroller that runs on the FreeRTOS operating system and is equipped with an lwIP TCP/IP library. It supports a wide range of protocols that allow users to control devices and applications with ease and efficiency.

1. Wi-Fi: This protocol allows users to connect to various Wi-Fi networks and control devices connected to them. The ESP32 supports Wi-Fi Direct, Wi-Fi Protected Setup (WPS), and Wi-Fi roaming.
2. Bluetooth: The ESP32 supports both Classic Bluetooth and Bluetooth Low Energy (BLE) protocols, which allow users to create Bluetooth connections with other devices and control them. The ESP32 can act as a Bluetooth master or slave, and it supports multiple simultaneous connections.
3. BLE: The ESP32 also supports BLE, which is a low-power protocol that is ideal for IoT applications. It allows users to create connections with other BLE devices and control them. The ESP32 can act as a BLE server or client.
4. MQTT: This protocol is used to connect to MQTT brokers and control devices connected to them. MQTT is a lightweight protocol that is ideal for IoT applications where low power consumption and bandwidth usage are critical.
5. HTTPS: HTTPS is a secure version of the HTTP protocol that is used to send and receive data over the internet. The ESP32 supports HTTPS, which allows users to securely send and receive data to and from web servers.
6. TCP/IP: The ESP32 supports the TCP/IP protocol, which is used to send and receive data over the internet. It also supports other protocols that run on top of the TCP/IP protocol, such as HTTP, FTP, and Telnet.
These protocols enable users to control devices connected to the ESP32 and send and receive data effectively. In addition to these protocols, the ESP32 also supports

various functions such as PWM, ADC, SPI, I2C, UART, and more, which allow users to control devices with precision and accuracy.
Overall, the ESP32 is a powerful and versatile microcontroller that supports a wide range of protocols, making it an ideal choice for IoT applications, smart home devices, robotics, and more.

- ***Advantages of esp32***

"ESP32 is a powerful MCU capable of meeting the requirements of various applications, and it has many features that make it a popular choice for many users in fields such as IoT, smart devices, robotics, medical devices, and many other applications. Some of the key features of ESP32 are:
1. Powerful and versatile: ESP32 has a powerful and versatile performance, making it suitable for a wide range of tasks and applications.
2. Multiple protocol support: ESP32 supports many different protocols, including Wi-Fi, Bluetooth, BLE, MQTT, HTTPS, TCP/IP, making it capable of communicating with many different devices and applications.
3. IoT application support: ESP32 supports IoT applications and has low power consumption, making it suitable for use in IoT applications that require low power consumption.
4. High performance: ESP32 operates at high speed and has powerful performance, making it suitable for many different applications.
5. Large number of control interfaces: ESP32 has a large number of control interfaces, including GPIO, SPI, I2C, UART, and others, allowing users to control devices accurately and efficiently.
6. Security support: ESP32 supports many security features, including encryption, digital signing, identity verification, and others, making it suitable for use in applications that require a high level of security.
7. Ease of use: ESP32 is easy to use, allowing users to customize it to meet their individual needs and easily connect it to other devices.
8. Free and open source: ESP32 is free and open source, allowing users to access the source code and modify and improve it.
Overall, ESP32 is a powerful and versatile MCU that supports many different protocols and allows users to control devices and applications easily and efficiently. ESP32 is a popular choice for many users in fields such as IoT, smart devices, robotics, medical devices, and many other applications, as it can meet the requirements of various applications, provide support for many different protocols and control interfaces, and support IoT applications and security, in addition to being easy to use and open source.

3.1.3: ESP32 vs ESP8266 vs Arduino

ESP32 VS ESP8266 & ARDUNIO UNO: -

The ESP32 is a cheap Wi-Fi module (and also ESP32 used to indicate board) which perfectly suited for DIY projects in the Internet of Things (IoT). When we are using ESP32 based boards with Arduino IDE (like, in our setup of Arduino UNO IDE guide) then the ESP32 is becoming "Arduino compatible" board. It is not what is supported by official Arduino by any means.

The Arduino UNO uses a 8-bit ATmega328P controller running at 16MHz with 2kRAM and 32kEPROM, while the ESP32 is a 32-bit CPU running at 160 or 240 MHz with 520kiB RAM and typically 1MB EPROM (external). It also includes Bluetooth and Wi-Fi with complete TCP: IP stack. Figure 3.1.3.1

Its peripherals include 2 sets of I²S pins, 2 sets of I²C pins (which makes interfacing with displays just easy, like we have shown in LCD setup and OLED setup guides).

It has 3 sets of UART, SD/SDIO/CE-ATA/MMC/eMMC host controller, Hall effect sensor and lot of features including cryptographic hardware acceleration for AES, SHA-2, RSA, ECC, low 5µA current consumption in deep sleep, capacitive touch sensor interrupt etc.

So, we can see that ESP32 worth it's $10 price with so many features. But ESP32 as Arduino has no official standard board. Commonly clones of ESP32 DEVKIT V1 DOIT board is used. Adafruit has good quality board but that is also not unique standard.

Although Arduino UNO is less powerful, it is still the basic board for newbies and easier to use. Using ESP32 as Arduino board need some getting used with Arduino. Arduino has well defined multiple analog input. It is not easy for the newbie to setup an ESP32 with Arduino IDE. In all ways, ESP32 is intended for the slightly advanced users who otherwise would code in C++. Getting started with the ESP32 as Arduino still not just following one official website. It is true that manufacturer of ESP32 as enough code to support Arduino, documentations etc.

SPECS/BOARD	ESP32	ESP8266	ARDUINO UNO
Number of Cores	2	1	1
Architecture	32 Bit	32 Bit	8 Bit
CPU Frequency	160 MHz	80 MHz	16 MHz
WiFi	YES	YES	NO
BLUETOOTH	YES	NO	NO
RAM	512 KB	160 KB	2 KB
FLASH	16 MB	16 MB	32 KB
GPIO PINS	36	17	14
Busses	SPI, I2C, UART, I2S, CAN	SPI, I2C, UART, I2S	SPI, I2C, UART
ADC Pins	18	1	6
DAC Pins	2	0	0

Connect ESP32 to Wi-Fi

Connecting ESP32 to Wi-Fi network requires several steps, and here are the basic steps to do so:

1. Connecting the antenna: Before connecting ESP32 to the Wi-Fi network, you need to connect the Wi-Fi antenna to the module. The antenna should be connected to the wireless antenna port on the ESP32.
2. Installing the Wi-Fi library: You need to install the Wi-Fi library in the development environment you are using. You can use Arduino IDE or Plat form IO for this purpose.
3. Connecting ESP32 to the Wi-Fi network: You need to connect ESP32 to the Wi-Fi network using the required SSID and password. You can do this using the Wi-Fi.begin() function in the Wi-Fi library.
4. Checking the connection: After connecting ESP32 to the Wi-Fi network, you need to check that the connection has been successful. You can do this using the Wi-Fi.status() function in the Wi-Fi library.
5. Using the Wi-Fi connection: After verifying that ESP32 has been connected to the Wi-Fi network, you can use the Wi-Fi connection in different applications. You can use the functions in the Wi-Fi library, such as WiFiClient() function to create a TCP/IP connection over the Wi-Fi network.

By following these steps, you can connect ESP32 to the Wi-Fi network and use the Wi-Fi connection in different applications. Note that connecting ESP32 to the Wi-Fi network requires correct settings for SSID and password, and you need to verify that the connection has been successful before using it in different applications.

```
void setup() {

  // initialize serial communication @ 9600 baud:
  Serial.begin(9600);
  Serial.begin(115200);
  pinMode(pulseLED, OUTPUT);
  pinMode(readLED, OUTPUT);



  WiFi.mode(WIFI_STA);
  WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
  Serial.println("---------------Connection");
  Serial.print("Connecting to : ");
  Serial.println(WIFI_SSID);
  while (WiFi.status() != WL_CONNECTED){
    Serial.print("."); delay(100);

  }
  Serial.println();
  Serial.print("Successfully connected to : ");
  Serial.println(WIFI_SSID);
  //Serial.print("IP : ");
  //Serial.println(WiFi.localIP());
  Serial.println("--------------");
  //----------------------------------------
```

Figure 3.1.3.2

```
  // Assign the api key (required).
  config.api_key = API_KEY;

  // Assign the RTDB URL (required).
  config.database_url = DATABASE_URL;

  // Sign up.
  Serial.println();
  Serial.println("---------------Sign up");
  Serial.print("Sign up new user... ");
  if (Firebase.signUp(&config, &auth, "", "")){
    Serial.println("ok");
    signupOK = true;
  }
  else{
    Serial.printf("%s\n", config.signer.signupError.message.c_str());
  }
  Serial.println("---------------");

  // Assign the callback function for the long running token generation task.
  config.token_status_callback = tokenStatusCallback; //--> see addons/TokenHelper.h

  Firebase.begin(&config, &auth);
  Firebase.reconnectWiFi(true);


Serial.println("Initializing...");
```

Figure 3.1.3.3

3.2: LM35 Temperature Sensor

LM35

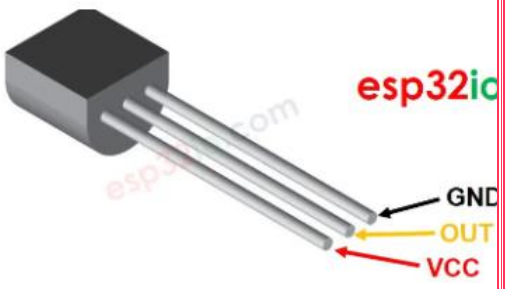
LM35 is a thermal sensor that is used to measure temperature. This sensor is characterized by its high accuracy and ease of use, and it can be used in many different applications.

LM35 is one of the most popular temperature sensors used in many electronic applications, where it is used to measure temperature with high accuracy. This sensor operates on direct voltage, where the red pin of LM35 is connected to the positive voltage and the black pin is connected to the ground, while the green pin is connected to the input for measuring the temperature. The voltage generated by LM35 is read directly and converted into a temperature value.

LM35 is characterized by high accuracy in measurement and ease of connection, where the measured temperature value is provided directly in the form of voltage

generated by the sensor without the need for any additional circuits. LM35 is used in many electronic applications, and its most important uses are:

1. Temperature control systems: where LM35 is used to determine the temperature and then control the temperature in different systems.
2. Alarm systems: where LM35 is used to determine the temperature and send an alarm signal when a specific temperature is exceeded.
3. Control of electronic devices: where LM35 is used to determine the temperature of electronic devices and then control them appropriately.
4. Robotics applications: where LM35 is used to determine the ambient temperature of the robot and then control the robot's behavior appropriately.



In this way, LM35 can be used in many different electronic applications, where it is used to measure temperature with high accuracy and in a simple and easy way.

Figure 3.2.1

LM35 temperature sensor has three Pin: -

- VCC pin: connect this pin to VCC (5V)
- GND pin: connect this pin to GND (Ov)
- OUT pin: This pin outputs voltage in proportion to the temperature value.

consists of The LM35

The LM35 consists of two main parts, namely:

- ***Temperature Sensor:***

It is a temperature sensitive part whose resistance changes with the change of temperature. This resistance is read using an electronic circuit to convert it into an electronic signal.

- **Signal conversion circuit:**

It is an electronic circuit that converts the sensor signal into an electrical signal that can be read by a computer or microcontroller.

LM35 & ESP32: -

- ***Simulation: -***

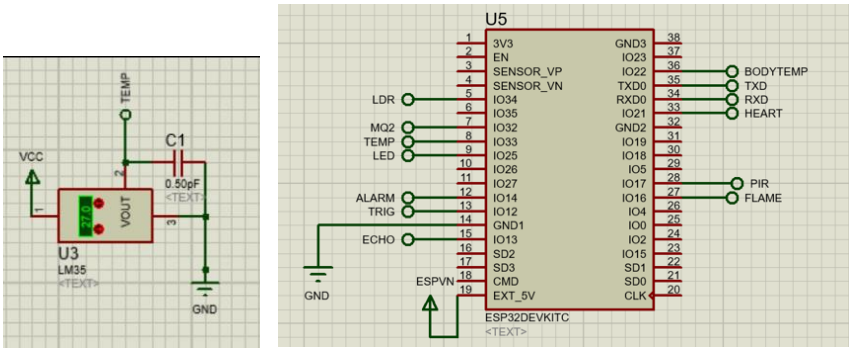


Figure 3.2.2

- ***Code***

LM35_TEMP.ino LM35_TEMP.ino

```
1   // Define the analog pin, the LM35's Vout pin is connected to
2   #define sensorPin A0
3
4   void setup() {
5     // Begin serial communication at 9600 baud rate
6     Serial.begin(9600);
7   }
8
9   void loop() {
10    // Get the voltage reading from the LM35
11    int reading = analogRead(sensorPin);
12
13    // Convert that reading into voltage
14    float voltage = reading * (5.0 / 1024.0);
15
16    // Convert the voltage into the temperature in Celsius
17    float temperatureC = voltage * 100;
18
19    // Print the temperature in Celsius
20    Serial.print("Temperature: ");
21    Serial.print(temperatureC);
22    Serial.print("\xC2\xB0"); // shows degree symbol
23    Serial.print("C  |  ");
24
25    // Print the temperature in Fahrenheit
26    float temperatureF = (temperatureC * 9.0 / 5.0) + 32.0;
27    Serial.print(temperatureF);
28    Serial.print("\xC2\xB0"); // shows degree symbol
29    Serial.println("F");
30
31    delay(1000); // wait a second between readings
```

Figure 3.2.3

3.3: Smoke Sensor (MQ2)

Figure 3.3.1

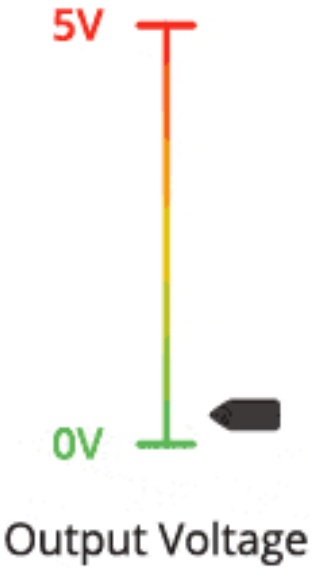
MQ2 Gas/Smoke Sensor

- ***MQ2 Gas/Smoke Sensor:*** **-** is a gas sensor that is used to detect gas and smoke leaks.
- The sensor works by detecting changes in the chemical concentrations of gases in the air, can be used to detect many different gases such as natural gas, methane, carbon dioxide, alcohol, ammonia, smoke, hydrogen, ethanol, benzene, aspartine, and toxic oxygen.

How Does a Gas Sensor Work?

- The sensor consists of several layers, where the first layer consists of a piece of metal that acts as an air heater. When the sensor is triggered, the metal layer is heated until it reaches a certain temperature.
- The second layer is located in the sensor, which is a layer of chemicals that are sensitive to various gases. When this layer is exposed to the surrounding air, changes in the concentration of the gases change its electrical resistance.
- The third layer is in the sensor, which is a layer of oxygen-sensitive chemicals. This layer produces an electrical signal that is inversely proportional to the oxygen concentration in the air.
- The electrical signal generated by the sensor is read using an electronic circuit, and the signal is converted into a digital or analog signal that can be readusingacomputeror microcontrollers.

Figure 3.3.2



- With the sensor, the concentrations of several gases in the air can be measured, such as natural gas, methane, carbon dioxide, alcohol, ammonia, smoke, hydrogen, ethanol, benzene, aspartine, and toxic oxygen.

Pins of MQ2 Gas/Smoke Sensor: -

- VCC: It is the power pin that is connected to the power supply to power the sensor.
- GND: It is the ground pin that connects to the ground of the power supply.
- AOUT: It is the analog signal pin produced by the sensor when changes in the chemical concentrations of gases are detected.

Figure 3.3.3

- DOUT: It is the digital signal pin

  produced by the sensor when changes in the chemical concentrations of gases are detected. This pin is used to notify other devices such as a microcontroller or computer that there is a gas leak**.**

MQ2 Gas/Smoke Sensor& ESP32: -

- ***Simulation: -***

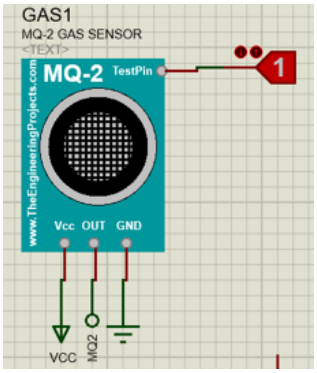
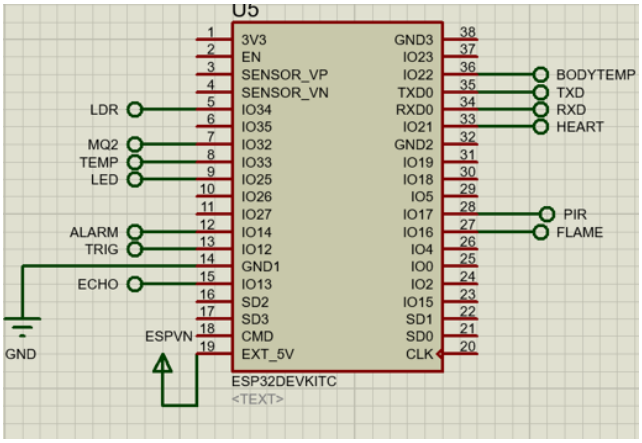


Figure 3.3.4



- ***Code : -***

MQ2_SMOKE.ino | MQ2_SMOKE.ino

```
/* Change the threshold value with your own reading */
#define Threshold 200

#define MQ2pin 0

float sensorValue;  //variable to store sensor value

void setup() {
  Serial.begin(9600); // sets the serial port to 9600
  Serial.println("MQ2 warming up!");
  delay(20000); // allow the MQ2 to warm up
}

void loop() {
  sensorValue = analogRead(MQ2pin); // read analog input pin 0

  Serial.print("Sensor Value: ");
  Serial.print(sensorValue);

  if(sensorValue > Threshold)
  {
    Serial.print(" | Smoke detected!");
  }

  Serial.println("");
  delay(2000); // wait 2s for next reading
}
```

Figure 3.3.4

3.4: Flame Sensor (yg1006)

FLAME Sensor: -

- Flame sensor is a device that is used to detect the presence of flame. This sensor is characterized by its high accuracy and speed in detecting flames, and it can be used in many different applications.
- Flame sensor is used in many applications, such as alarm system for fire detection in homes and buildings and alarm system for industrial equipment and utilities.

Flame Sensor (yg1006) & ESP32: -

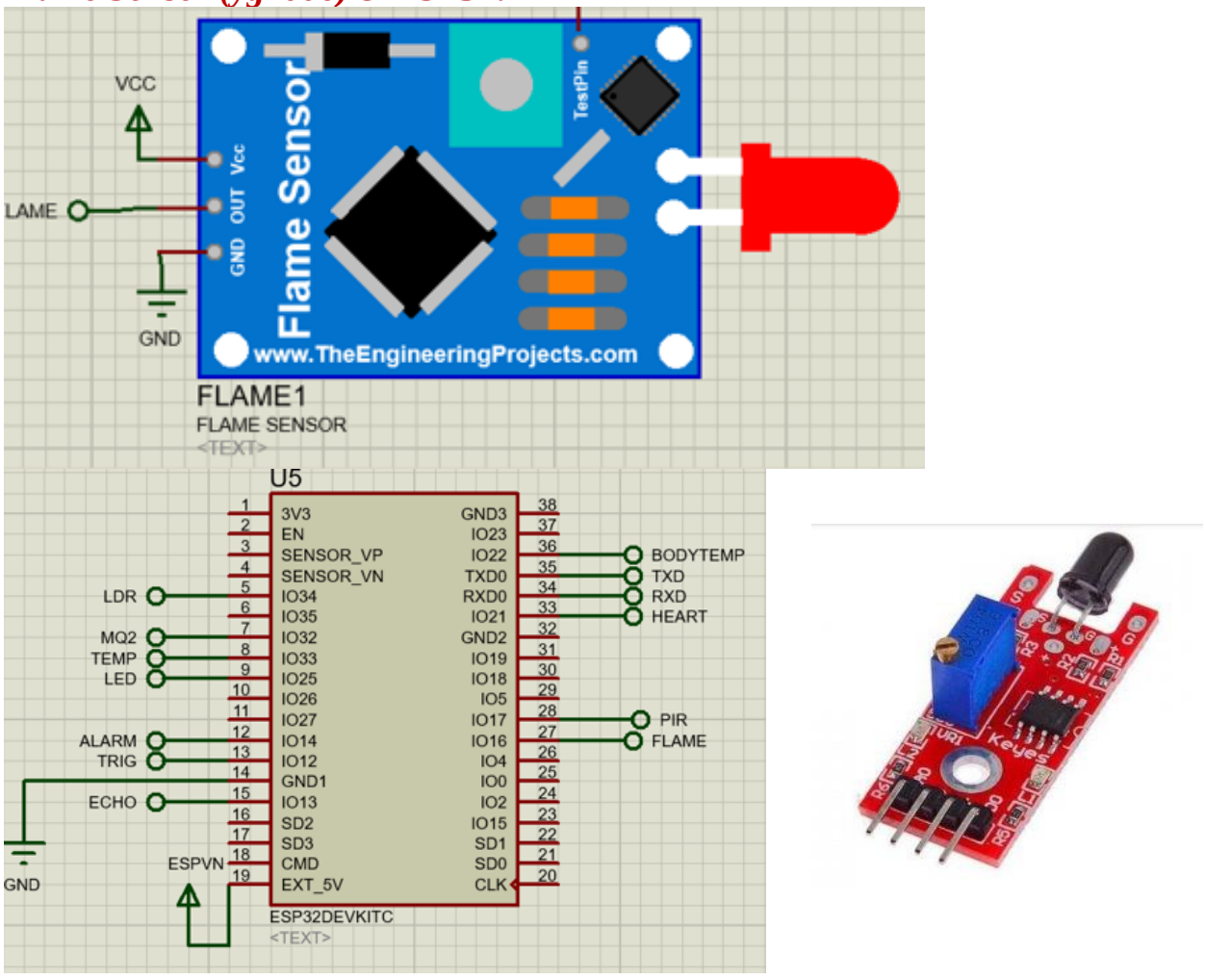


Figure 3.4.1

CODE: -

```
/* Flame Sensor analog example.
To test view the output, point a serial monitor such as Putty at your arduino.
*/

// lowest and highest sensor readings:
const int sensorMin = 0;     // sensor minimum
const int sensorMax = 1024;  // sensor maximum

void setup() {
  // initialize serial communication @ 9600 baud:
  Serial.begin(9600);
}
void loop() {
  // read the sensor on analog A0:
  int sensorReading = analogRead(A0);
  // map the sensor range (four options):
  // ex: 'long int map(long int, long int, long int, long int, long int)'
  int range = map(sensorReading, sensorMin, sensorMax, 0, 99);

  // range value:

  if (range < 50){    // A fire closer than 1.5 feet away.
    Serial.println("** Close Fire **");
  }  else if (100 > range > 50){    // A fire between 1-3 feet away.
    Serial.println("** Distant Fire **");
  }  else if (range > 100){    // No fire detected.
    Serial.println("No Fire");
  }
  delay(500);  // delay between reads
}
```

Figure 3.4.2

3.5: GPS Module (Ublox NEO-7m)

Implementation of sensors & devices on the controller

Ublox NEO-6M GPS Module

Figure 3.5.1

Sensor hardware

First NEO-6M GPS Chip

The sensor can track up to 22 satellites over 50 channels and its sensitivity -161db and consumes 45mA current.

It can also perform or update 5 locations in a second

Power saving mode (PSM) is one of its features that make it consumes only 11mA current

Figure 3.5.2

Here is the specifications

• Receiver Type	50 channels, GPS L1(1575.42Mhz)
• Horizontal Position Accuracy	2.5m
• Navigation Update Rate	1HZ (5Hz maximum)
• Capture Time	Cool start: 27sHot start: 1s
• Navigation Sensitivity	-161dBm
• Communication Protocol	NMEA, UBX Binary, RTCM
• Serial Baud Rate	4800-230400 (default 9600)
• Operating Temperature	-40°C ~ 85°C
• Operating Voltage	2.7V ~ 3.6V

- Operating Current 45mA
- TXD/RXD Impedance 510Ω

Second position fix led indicator

There is an LED on the NEO-6M GPS module that indicates the status of the 'Position Fix':

- No blinking – it is searching for satellites.
- Blink every 1s – Position Fix is found (the module can see enough satellites)

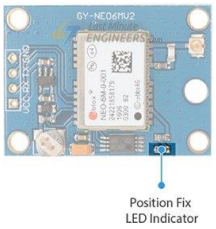


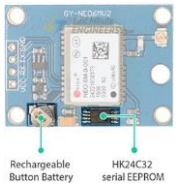
Figure 3.5.3

Third 3.3V LDO Regulator

This module comes with MICREL's MIC5205 Ultra-Low Dropout 3V3 regulator. The logic pins are also 5-volt tolerant, so we can easily connect it to Arduino or any 5V logic microcontroller without using a logic level converter.

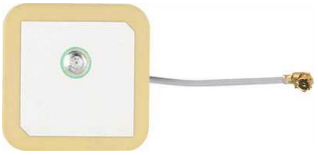
Finally Battery & EEPROM

wo Wire Serial EEPROM. It is 4KB in size and is connected via I2C to the NEO-6M chip.
The module also houses a rechargeable button battery that acts as a super-capacitor.



Another thing but it comes with the module is the Antenna

The module comes with -161 dBm sensitivity patch antenna for receiving radio signals from GPS satellites.
It is connected to the U.FL connector in the module



GPS Pinout

1- GND : is a ground pin connected to the GND pin to the microcontroller.
2- TX : pin is used for serial communication (transmitter).
3- RX : pin is used for serial communication (receiver).
4- VCC : supplies power to the module. It can be connected directly to the 5V

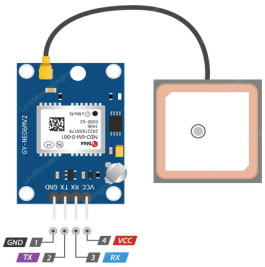


Figure 3.5.4

GPS with ESP32

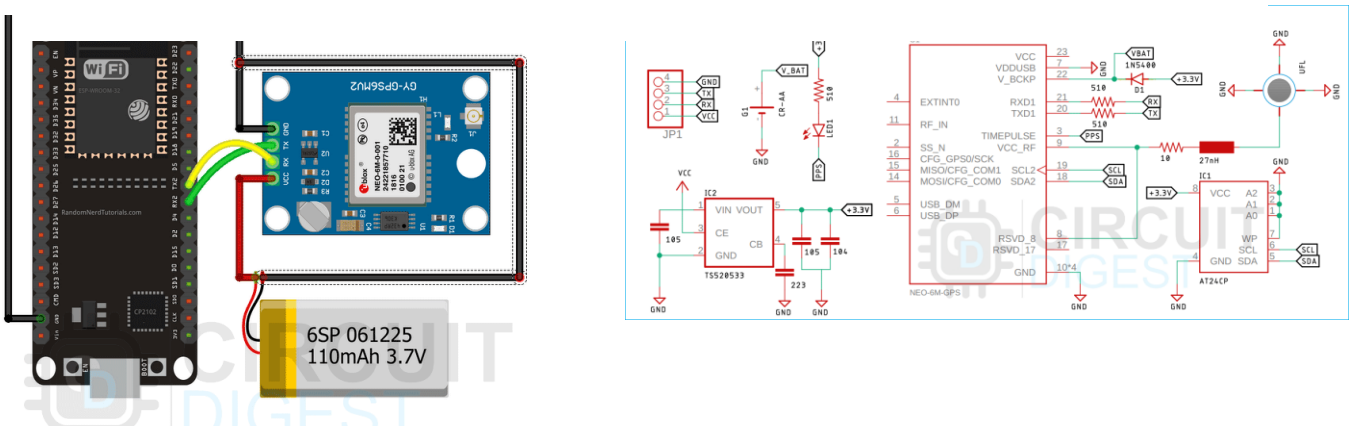


Figure 3.5.5

Code shots

Tiny library is used in gps and you should install it in arduino

```
#include <TinyGPSPlus.h>

//library for gps

TinyGPSPlus gps;

void setup() {

  Serial.begin(9600);

  Serial2.begin(9600);

  delay(3000);

}
```

Figure 3.5.6

Next we have our **updateSerial()** function. This function is a **loopback** between the **UART1** and **UART2** so that we can monitor the incoming data out of the serial monitor window.

```
void updateSerial()

{

 delay(500);

 while (Serial.available())

 {

  Serial2.write(Serial.read());//Forward what Serial received to Software Serial Port

 }

 while (Serial2.available())

 {

  Serial.write(Serial2.read());//Forward what Software Serial received to Serial Port

 }
```

Figure 3.5.7

Figure 3.5.7

GPS and the application

After all of this the GPS will be connected to a firebase and then the location will be shown in the mobile application

3.6: Ultrasonic Sensor (HC-SR04)

Ultrasonic sensor

Figure 3.6.1

Sensor hardware

It consists of two ultrasonic transducers.
One acts as a transmitter that converts the electrical signal into 40 KHz ultrasonic sound pulses. The other acts as a receiver and listens for the transmitted pulses. When the receiver receives these pulses, it produces an output pulse whose width is proportional to the distance of the object in front.
This sensor provides excellent non-contact range detection between 2 cm to 400 cm (~13 feet) with an accuracy of 3 mm.
Since it operates on 5 volts, it can be connected directly to an Arduino or any other 5V logic microcontroller.

Here are the specifications:

- Operating Voltage — DC 5V
- Operating Current — 15mA
- Operating Frequency — 40KHz
- Max Range — 4m
- Min Range — 2cm
- Ranging Accuracy — 3mm
- Measuring Angle — 15 degree
- Trigger Input Signal — 10µS TTL pulse
- Dimension — 45 x 20 x 15mm

Ultrasonic Sensor pinout

1- VCC supplies power to the ultrasonic sensor.

2- Trig (Trigger) pin is used to trigger ultrasonic sound pulses. By setting this pin to HIGH for 10µs, the sensor initiates an ultrasonic burst.
3- Echo pin goes high when the ultrasonic burst is transmitted and remains high until the sensor receives an echo, after which it goes low. By measuring the time the Echo pin stays high, the distance can be calculated.
4- GND is the ground pin. Connect it to the ground of the Arduino.

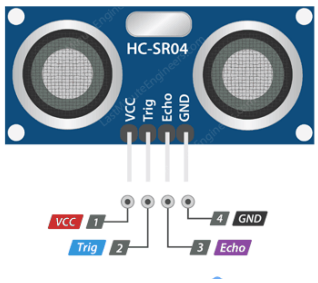
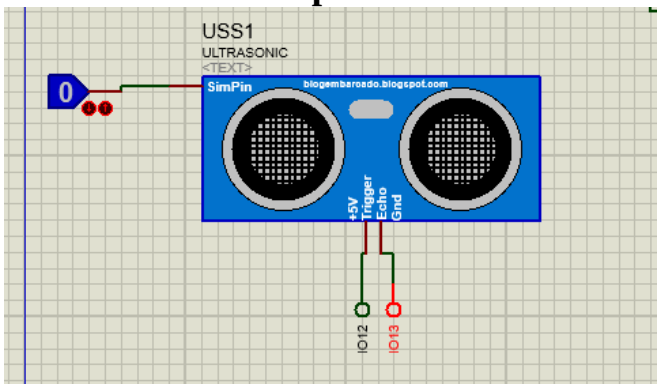


Figure 3.6.2

Ultrasonic with esp32



Code capture

```
// Define the pins for the ultrasonic sensor and the buzzer
const int trigPin = 12;
const int echoPin = 13;
const int buzzerPin = 10;

// Define the maximum distance to trigger the buzzer
const int maxDistance = 35; // in centimeters

void setup() {
  // Initialize the serial communication for debugging
  Serial.begin(9600);

  // Configure the pins for the ultrasonic sensor and the buzzer
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
  pinMode(buzzerPin, OUTPUT);
}

void loop() {
  // Trigger the ultrasonic sensor to send a pulse
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
```

```
  // Measure the duration of the pulse and calculate the distance
  long duration = pulseIn(echoPin, HIGH);
  int distance = duration * 0.034 / 2;

  // Print the distance for debugging
  Serial.print("Distance: ");
  Serial.print(distance);
  Serial.println(" cm");

  // If the distance is greater than the maximum distance, turn on the buzze
  if (distance > maxDistance) {
    digitalWrite(buzzerPin, HIGH);
  } else {
    digitalWrite(buzzerPin, LOW);
  }

  // Wait for a short amount of time before measuring again
  delay(100);
}
```

Figure 3.6.3

Ultrasonic with our bike

We put our sensor at the end of the bike or in the bottom and calculated the distance which is 35 cm. Then if any one carries the bike then the alarm will be on (buzzer)
And it will be known that is there's something wrong.

Figure 3.6.4

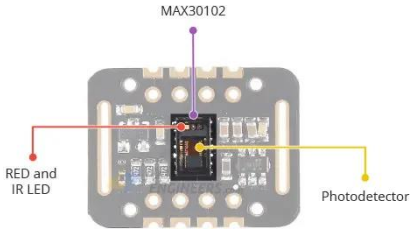
3.7: Heart Rate Monitoring Sensor (MAX30102)

Heart rate sensor (MAX30102)

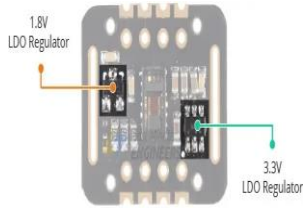
Figure 3.7.1

Sensor hardware

The heart rate sensor has a chip called (MAX30102) which has a photo detector and two leds (Red led and IR led)



It also has two regulators (1.8 reg for IC and 3.3 reg for Leds)



How it works

first we should know that the sensor can measure three things :

- Pulse oximetry
- Heart rate
- Temperature

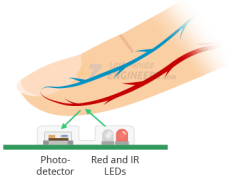


Figure 3.7.2

Second to know them the user puts his finger in the sensor so leds shining on the finger then reflected to the photo detector and from the change in the reflection waves the heart rate and SpO2 can be measured

The heart rate measurement

The oxygenated hemoglobin (HbO2) in the arterial blood has the characteristic of absorbing IR light. The redder the blood (the higher the hemoglobin), the more IR light is absorbed. As the blood is pumped through the finger with each heartbeat, the amount of reflected light changes, creating a changing waveform at the output of the photodetector.

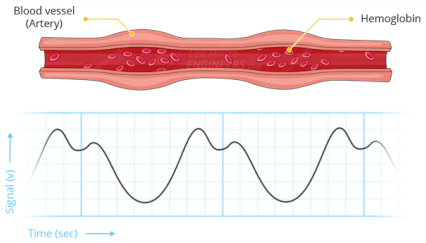


Figure 3.7.3

As you continue to shine light and take photodetector readings, you quickly start to get a heart-beat (HR) pulse reading

Pulse oximetry

Pulse oximetry is based on the principle that the amount of RED and IR light absorbed varies depending on the amount of oxygen in your blood. The following graph is the absorption-spectrum of oxygenated hemoglobin (HbO2) and deoxygenated hemoglobin (Hb).

As you can see from the graph, deoxygenated blood absorbs more RED light (660nm), while oxygenated blood absorbs more IR light (880nm).

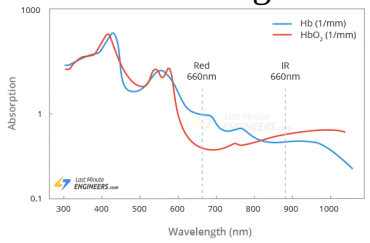


Figure 3.7.4

By measuring the ratio of IR and RED light received by the photodetector, the oxygen level (SpO2) in the blood is calculated.

Temperature

Heart rate sensor has on chip temperature sensor so it can easily calculate the temperature and it comes with accuracy 1 c
And its range is from -40 t0 +85 c

Heart rate Pinout

1-GND is the ground
2-SCL : is the I2C clock pin
3-SDA : is the I2C data pin
4-VIN : is the power pin.
5-INT : for interrupt
6-IRD : if you want to drive IR led
7-RD : if you want to drive RED led

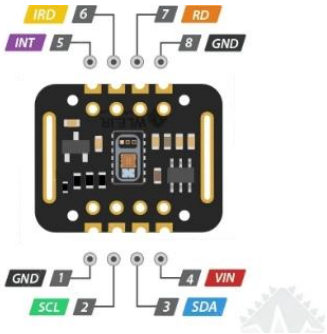
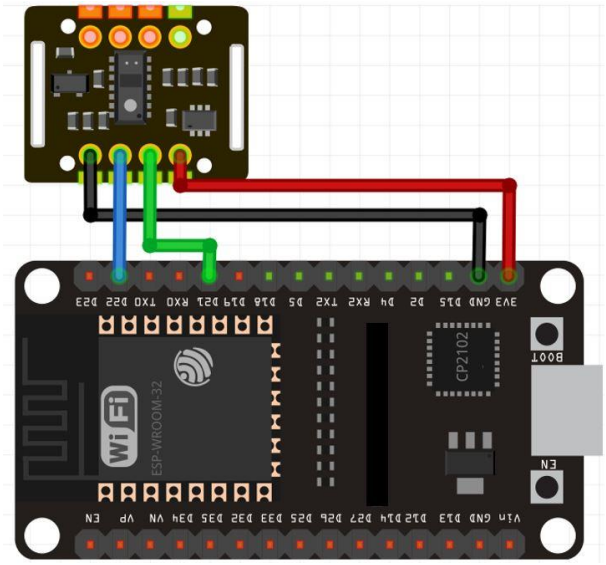


Figure 3.7.5

Heart rate with the esp32



Code capture

```
#include <Wire.h>
#include "MAX30100_PulseOximeter.h"

#define I2C_SDA 21
#define I2C_SCL 22

PulseOximeter pox;
uint32_t lastBeat = 0;
float beatsPerMinute;
int32_t spo2;
float temperature;

void setup() {
  Serial.begin(115200);
  Wire.begin(I2C_SDA, I2C_SCL);

  if (!pox.begin()) {
    Serial.println("Failed to initialize pulse oximeter");
    while (1);
  }

  pox.setIRLedCurrent(MAX30100_LED_CURR_7_6MA);
}
```

Figure 3.7.6

```
void loop() {
  pox.update();

  if (millis() - lastBeat > 10000) {
    lastBeat = millis();
    beatsPerMinute = pox.getHeartRate();
    spo2 = pox.getSpO2();
    temperature = pox.getBodyTemperature();

    Serial.print("Heart Rate: ");
    Serial.print(beatsPerMinute);
    Serial.println(" bpm");

    Serial.print("SpO2: ");
    Serial.print(spo2);
    Serial.println(" %");

    Serial.print("Temperature: ");
    Serial.print(temperature);
    Serial.println(" C");
  }
}
```

Figure 3.7.7

Last thing with heart rate

Every reading it get will be visible in our application
As temperature, SpO2, heart rate

Figure 3.7.8

3.8: LDR (Flashlight + Red Backlight)

A Light Dependent Resistor (LDR) is a type of resistor that changes its resistance based on the intensity of light it is exposed to. In darkness, its resistance is very high, while in bright light its resistance is very low. This change in resistance makes it best for light sensing projects.

The ESP32 analog pins convert the incoming voltages to an integer between 0 and 4095. This integer value is mapped against the analog input voltage from 0V to 3.3V which is

by default the ADC reference voltage in ESP32. This value is read using the Arduino analogRead() function from LDR.

The ESP32 has a built-in analog-to-digital converter (ADC) that can measure the voltage across the LDR and convert it into a digital signal that can be processed by the microcontroller. Using this signal ESP32 determines the resistance of the LDR, which is proportional to the light intensity.

Photons or light particles play a crucial role in the operation of LDRs. When light falls on the surface of an LDR, photons are absorbed by the material, which then frees electrons in the material. The number of free electrons is directly proportional to the intensity of light, and the more electrons that are freed, the lower the resistance of the LDR becomes.

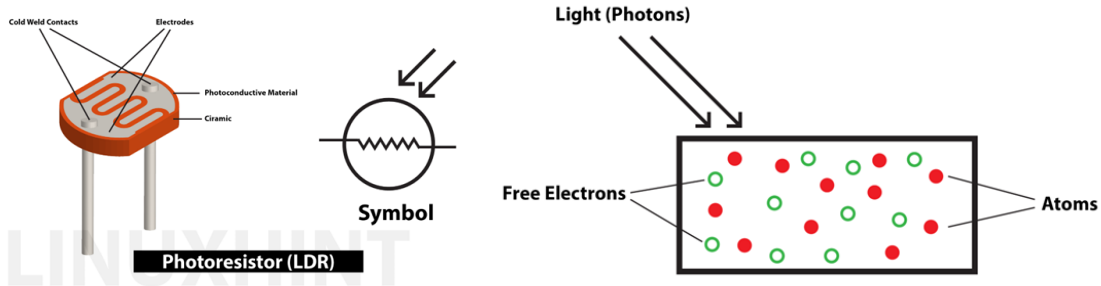


Figure 3.8.1

Simulation: -

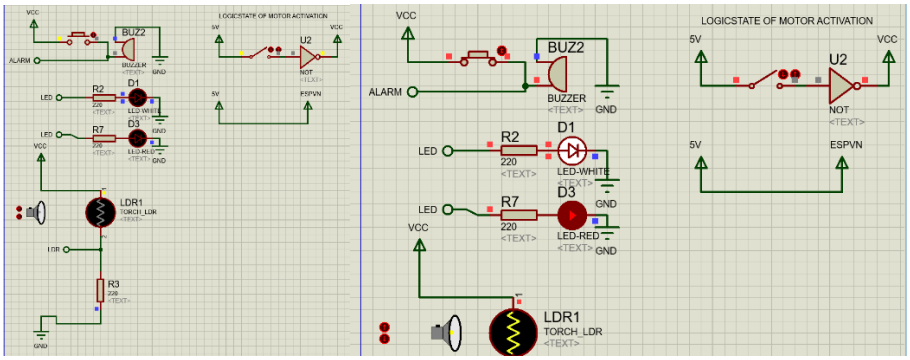


Figure 3.8.2

Code: -

DOIT ESP32 DEVKIT V1

sketch_feb13a.ino | debug_custom.json

```
int LDR_Val = 0;      /*Variable to store photoresistor value*/
int sensor =34;       /*Analogue Input for photoresistor*/
int led= 25;          /*LED output Pin*/
void setup() {
    Serial.begin(9600);      /*Baud rate for serial communication*/
    pinMode(led, OUTPUT);    /*LED Pin set as output */
}
void loop() {
    LDR_Val = analogRead(sensor);    /*Analog read LDR value*/
    Serial.print("LDR Output Value: ");
    Serial.println(LDR_Val);    /*Display LDR Output Val on serial monitor*/
    if(LDR_Val > 100) {        /*If light intensity is HIGH*/
        Serial.println(" High intensity ");
        digitalWrite(led,LOW); /*LED Remains OFF*/
    }
    else {
     /*Else if Light intensity is LOW LED will Remain ON*/
        Serial.println("LOW Intensity ");
        digitalWrite(led,HIGH);  /* LED Turn ON LDR value is less than 100*/
    }
    delay(1000);      /*Reads value after every 1 sec*/
}

```

Output | Serial Monitor

Building sketch | Ln 23, Col 1 | UTF-8 | DOIT ESP32 DEVKIT V1 on COM3 | 2

Figure 3.8.3

Application: -

- Light up flashlight and backlight when dark automatically and shut them off when there is no need in light.
- Could detect fire in a very dark place.

Figure 3.8.4

3.9: PIR Motion Sensor (HC-SR501)

How does a PIR sensor work?

All objects, including the human body, at temperatures above absolute zero (0 Kelvin / -273.15 °C) emit heat energy in the form of infrared radiation. The hotter an object is, the more radiation it emits. This radiation is not visible to the human eye because it is emitted at infrared wavelengths. The PIR sensor is specifically designed to detect such levels of infrared radiation.

A PIR sensor consists of two main parts:

1. A pyroelectric sensor, which you can see in the image below as a round metal with a rectangular crystal in the center.
2. A special lens called a fresnel lens which Focuses the infrared signals on the pyroelectric sensor.

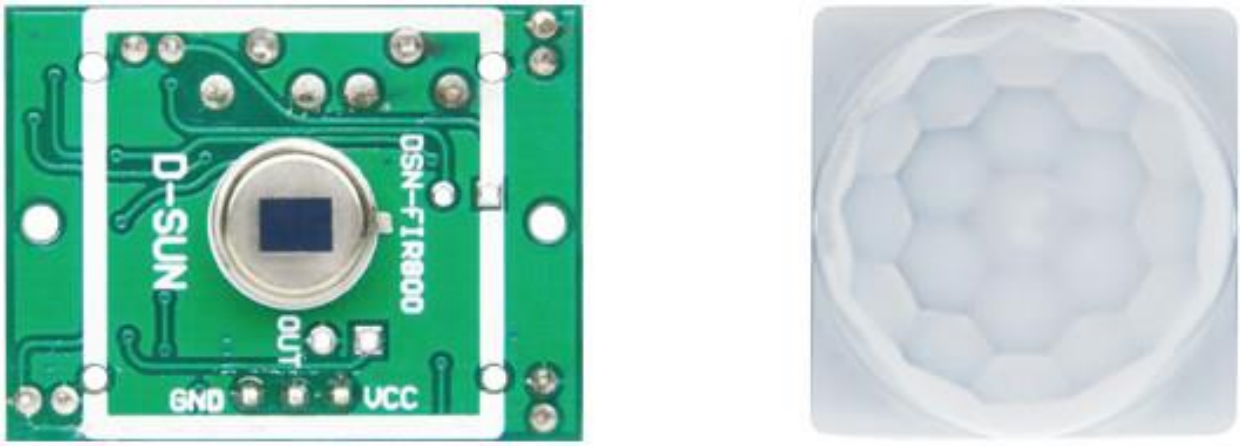


Figure 3.9.1

Figure 3.9.2

Specifications: -

Operating Voltage ..... 4.5 – 20V (typically 5V)

Maximum Current Draw..... < 2mA

Time Delay .....~ 1 sec to 3 min

Detection Distance ..... 3 – 7 meters (9 – 21 feet)

Detection Angle..... 120 degrees (typically)

Simulation: -

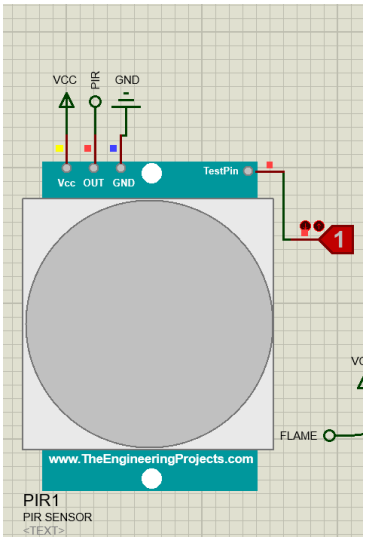


Figure 3.9.3

Code: -

PIR_Motion_Sensor.ino

```
const int pirPin = 2; // PIR sensor input pin
int motionCounter = 0; // Counter variable to track continuous motion
unsigned long motionTime = 0; // Time when motion was first detected

void setup() {
  pinMode(pirPin, INPUT);
  Serial.begin(9600); // Initialize serial communication for debugging
}

void loop() {
  int motionDetected = digitalRead(pirPin);
  if (motionDetected == HIGH) { // Motion detected
    if (motionCounter == 0) { // First time motion is detected
      motionTime = millis(); // Save the current time
      Serial.println("Motion detected!"); // Debugging message
    }
    motionCounter++; // Increment the motion counter
    if (motionCounter >= 180) { // 180 x 1 sec = 180 sec = 3 min
      Serial.println("Continuous motion detected for 3 minutes!"); // Debugging message
      // Do something here, e.g. trigger an alarm or activate a security system
    }
  } else { // No motion detected
    motionCounter = 0; // Reset the motion counter
  }
  delay(1000); // Delay for 1 second
}

```

Figure 3.9.4

Application: -

It is used in this project as one of the layers in the layered structure of security features added and acts only as a warning sign for suspicious activity near the bike.

3.10: Lithium Battery to USB Charging Module

Specifications: -

LED lights show the power, non-working state of intelligent automatic shutdown;

Dual USB output.

Built-in lithium battery protection IC, over-current, over-voltage, under-voltage protection.

18650 Lithium li-ion battery dual micro usb booster board parameter.

Input port: MicroUSB.

Input requirements: 5V constant voltage power supply can be done to charge the input power supply, the most matching charger for 5V1A above (Mobile Charger)

Output port: USB

Output parameters: 5V1A/5V2A

Size : 8 X 32 X 69 mm (H*W*L)

Application: -

It is used in this project to ensure that the user's mobile phone is charged or have enough charge to work properly and sustain the project's application without shutting down due to low battery and forcing the user to lose the network's features.

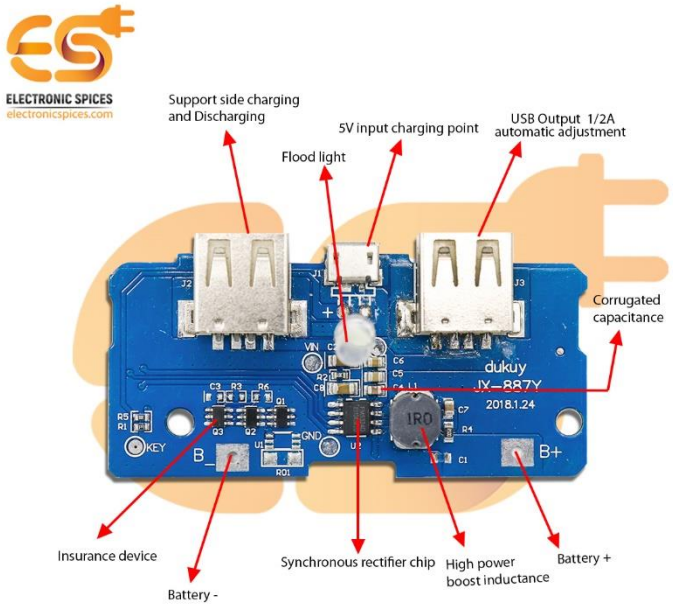


Figure 3.10.1

3.11: DC-DC Converter (from 48v to 5v) (XL7015)

XL7015 DC – DC buck type module is A highly efficient, high pressure step-down DC – DC converter, provides the highest 0.8 A output current capability, low ripple, excellent linear regulation and load regulation, up to A maximum of 80 v input voltage, more than 2596 modules, etc. Chip built-in various protection mechanisms. Can be applied to electric vehicle controller, etc.

Specifications: -

1. 5V to 80V input voltage range wide;
2. The output voltage from 5V to 20V adjustable;
3. The minimum pressure drop 1V.
4. 0.8A biggest switch current;
5. Recommend the output power is less than 7W;
6. Shut off built-in overheating protection, output short circuit protection, over current protection;
7. Good linear with load regulation;

Figure 3.11.1

Wiring instructions: -

1. VIN: input voltage (5V – 80V);
2. The GND: input voltage side;
3. The VOUT: output voltage (5V – 20V);
4. The GND: output voltage to end;

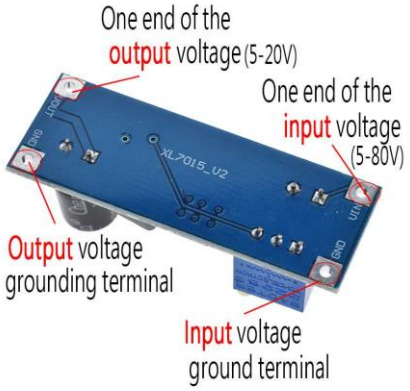


Figure 3.11.1

Application: -

Used in this project to convert the main battery's voltage from 48V (54V no load voltage) to 5V and connect it to power rail of sensor network and hence, making the 48V battery the powering heart of the whole project.

Part Two: User Interface

Chapter 4: Mobile Application

4.1: Features

In our project, The E-bike's speed and power, voltage, battery consumption, pedal assist level, weather condition, in addition to user's vital signs such as pulse rate, oxygen rate in Blood (SpO2), body temperature is being displayed on a mobile application.

The Mobile Application in our project is planned for monitoring parameters stated above as well as navigation, pedal lock control, GPS tracking of the bike in case of theft, and toggle control of smart features (LDR Flashlight for example).

The Application is designed by Figma. Figma is a cloud-based design tool that is like Sketch in functionality and features.

4.2: UI/UX Design

What is UX?

- User experience
- What a person feels as he experiences a product or service.
- Important role in all kinds of marketing.
- Bridge between a product and its targeted audience.

- Wireframe one of the most crucial steps which involves visualizing the skeleton of digital applications and is a layout of a product that demonstrates what interface elements will exist on key pages. It is a critical part of the interaction design process.

What is UI?

- User interface.
- How the people interact with other products/service.
- Used in web & mobile applications.
- A bridge between a human being and system.

UI/UX beyond Web & Apps?

- Design centered approach has become prime focus for technology driven company, because a good design solves problems.
- As the trend continues more and more industries are opting for design centered approaches for better handling of customers and maintaining higher standards of service.
- A good design is invisible. Even five year olds can handle it.

Design by FIGMA

- It's free.
- Working together simultaneously.
- Easy to share files.
- An all-round package for developing design prototypes.

Logo Screen:

It is the Startup screen when you first open the application

Figure 4.2.1

Sign up Screen:

It is to register user information for the first time

and create a new account into the application database

Sign in Screen:

It is to log in using user information registered from before.

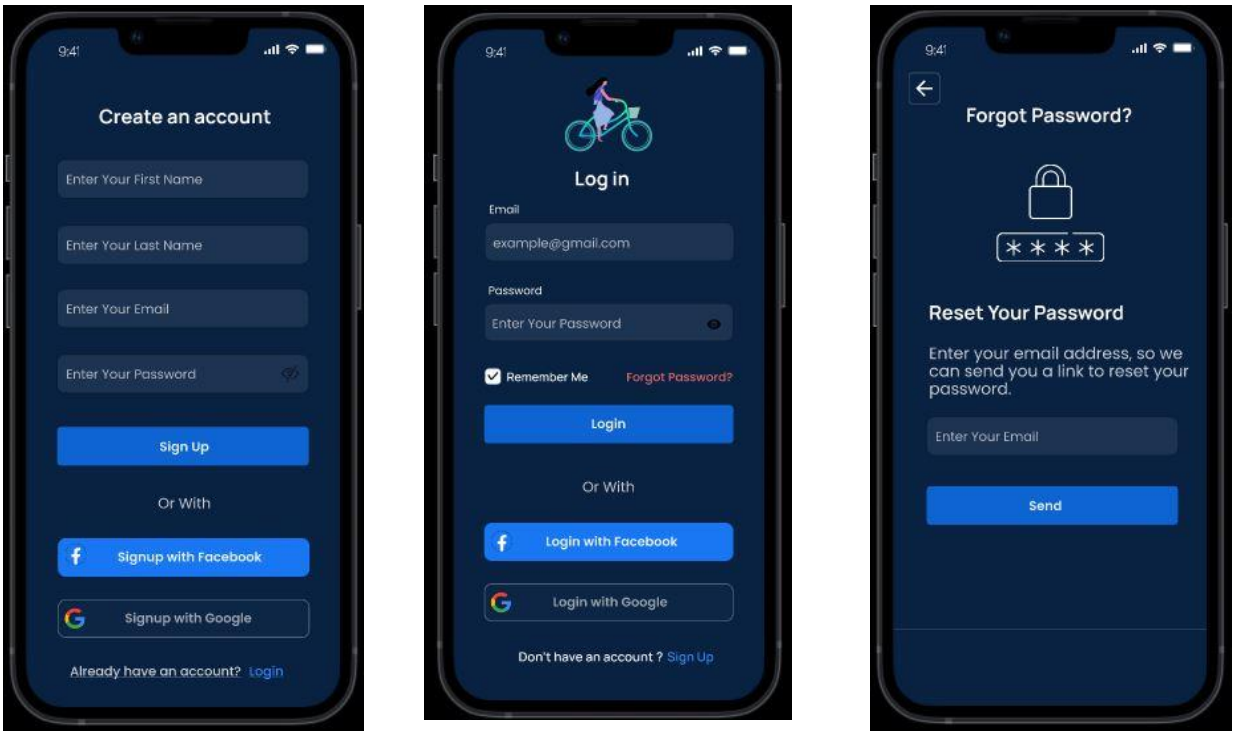


Figure 4.2.2

Home Screen:

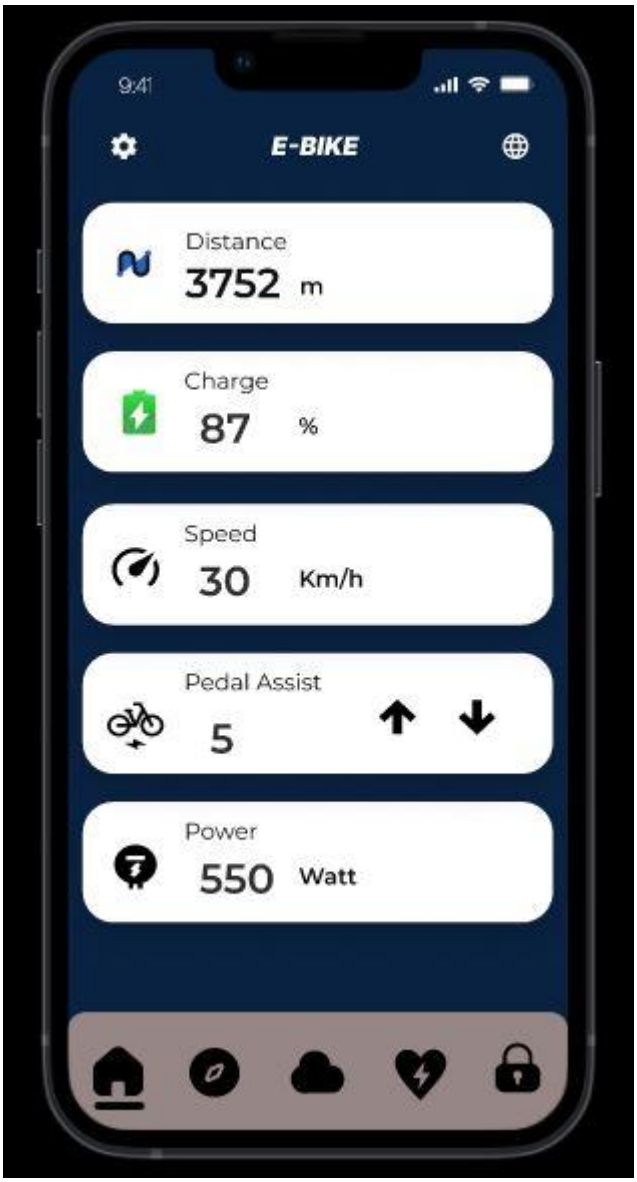


Figure 4.2.3

First Card: Distance

To measure the distance travelled by the electrical bike and present it in the distance card.

Second Card: Charge

To measure the power charge of the battery and present it in the charge card.

Third Card: Speed

To measure the speed of the electrical bike and present it in the speed card.

Fourth Card: Pedal Assist

As one of its levels from 0 to 5 is activated, the motor kicks in while you are pedaling. (Every level is optimized to output 0%-100% of the motor's top speed with a step of 20%)

Fifth Card: Power: To measure the power that comes out of the motor while you drive the electrical bike and present it in the power card.

Home Screen (Sensor ver.)

First Card: Distance:

To measure the distance travelled by the electrical bike and present it in the distance card.

Second Card: Battery Temperature:

To measure the Teamperature of battery and monitor it to prevent overheating and present it in the temp card.

Third Card: Smoke Card:

To measure the CO2 gas present in fire smokes nearby and monitor it to prevent fire and present it in the smoke card.

Fourth Card: Flame Card:

To precieve flame present in fires nearby and monitor it to prevent fire and present it in the flame card.

Figure 4.2.4

Fifth Card: Flashlight Card:

To manually control the front flashlight and red backlight when the LDR not needed or the flashlight is needed above light intensity threshold.

Website icon:

To transfer from application to website page of electrical bike.

Figure 4.2.5

Navigation Screen:

Navigation Is a real-time map of the current location and step-by-step directions to a requested destination.

Figure 4.2.6

Weather Screen:

Figure 4.2.7

Weather Card:

Its present the day and date and the temperature of the weather in card.

Also, present details of the weather like:

- Haze (wind speed) in km/h.
- Percentage of the cloud.
- Time of sunrise and sunset.
- Temperature of the day.
- Humidity of the day.

Could also give out warnings to users on a raining day to advise not to ride on that day or wait for a later time after rain.

Heath Screen:

Heart Rate Card:

To measure the heart rate of the body and present it in the card of the application.

Body Temperature Card:

To measure the temperature of the body and present it in in the card of the application.

Oxygen Percent in Blood Card:

To measure oxygen percent in body and present it in the card of the application.

Figure 4.2.8

GPS and Pedal Lock Screen

GPS:

If the electrical bike is lost or somehow stolen from owner the GPS tells the owner where the electrical bike is by GPS sensor.

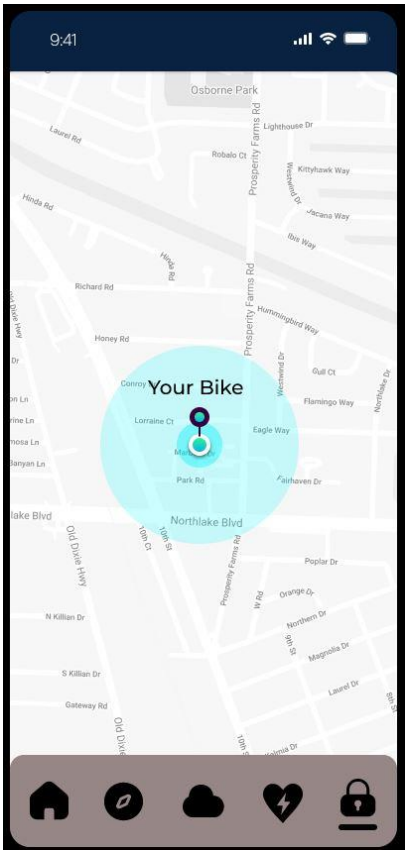


Figure 4.2.9

Settings Screen

Account:

To present the information of the user.

Ride History:

To present the history of your past electrical bike rides and the distance travelled, time elapsed, points acquired.

Notification:

It's a toggle. If you want to show notification while not using the application

Log out:

For logout of application account.

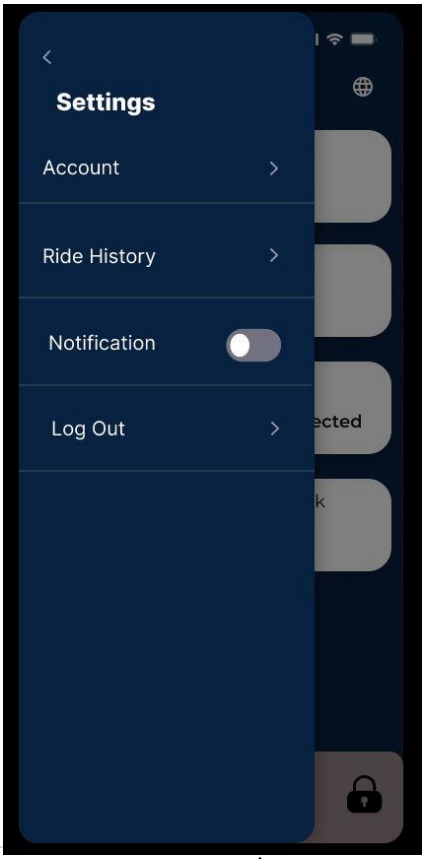


Figure 4.2.10

Account Screen:

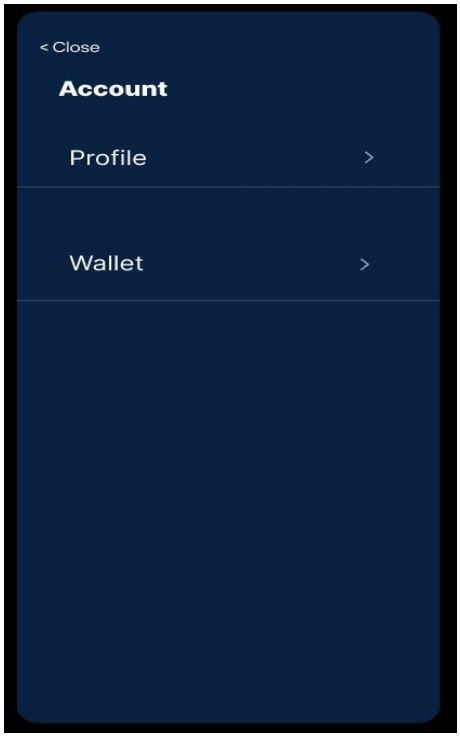


Figure 4.2.11

Profile:

Present information about the user who rides the electrical bike.

Payment:

Present the credit card used and the balance of the user.

Wallet:

Present the balance of the points you make by riding the electrical bike, points can be redeemed for electrical bike accessories from website store.

Profile Screen:

Figure 4.2.12

First:

We have the photo of the user.

Second:

The name of the user.

Third:

E-mail of the user.

Last:

You can change the account password of the user if you want to by the application.

Ride History Screen:

If you want to search any day you drive the electrical bike will present the day and the time of ride, then if you click on it will present the total distance of your ride and the duration of ride.

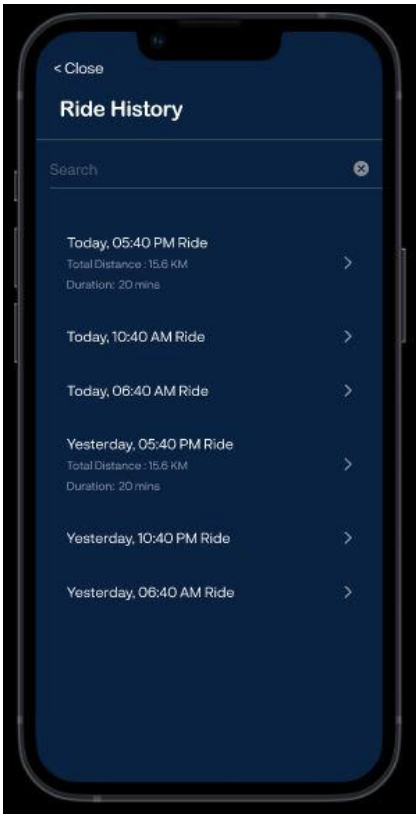


Figure 4.2.13

Warning Screen:

If the battery high temperature is too high, then the application will present a warning notification to the user.

Another Example:

If the heart rate and body temperature is too high, then the application will present a warning notification to the user.

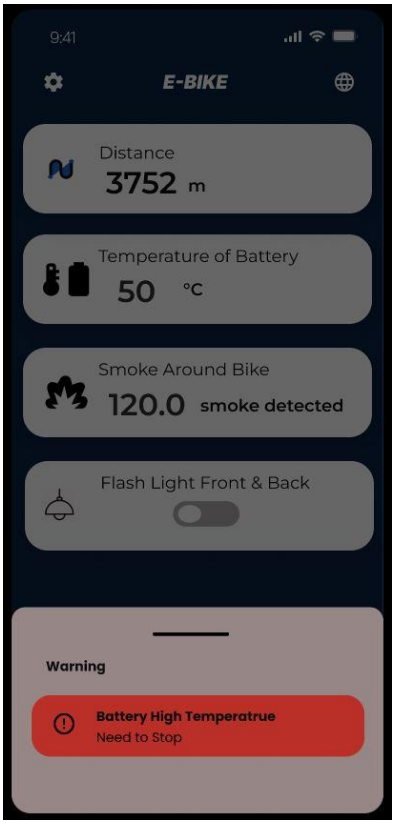


Figure 4.2.14

4.3: Front-End Development

Mobile Application

Introduction

Smartphones is essentially needed in daily personal use of person's day and night And each one has at least 1 device whether its Android or Ios , so considering this we can use devices for almost everything we can think of ,using esp32 Wi-Fi client and integrating it with firebase I build mobile application to help keep monitoring of health and take care of the user.

Flutter

Flutter is a framework based on Dart programming language created by google, and it makes it look easy to build a cross-platform application with the same code for Mobile android, Ios, Linux and windows, and its popularity become bigger last 5 years due to fast development cycles and flexible UI, native like performance.

Flutter Key Features

Dart Programming Language: Flutter uses the Dart programming language, which is known for its simplicity, fast execution, and reactive programming capabilities. Dart offers features like a ahead-of-time (AOT) compiler for optimized production performance.

Beautiful and Customizable UI: Flutter offers a rich set of customizable UI widgets, enabling developers to create visually appealing and highly interactive user interfaces. Flutter's widgets are designed to follow Material Design and Cupertino (iOS) guidelines, making it easy to create native-looking interfaces on different platforms.

Fast Development and Hot Reload: Flutter provides a fast development cycle with its "hot reload" feature. Changes made to the code are quickly reflected in the app, allowing for real-time experimentation, iteration, and debugging without restarting the app or losing its current state.

E-Bike App

The application provides functionality for keep tracking of the rider health and the bike safety from smoke or fires, traffic jam with great user interface and experience.

Application Objectives

- Monitoring health of driver using sensors & esp32 integrated with firebase.
- display current weather to avoid the bad weather.
- implementing Google maps for better routes for long travel distance
- encouraging people to bicycling to keep their bodies fit by implementing points rewarding.

Programs and Package Used

- **Android Studio along with Flutter Framework**: visual studio for editing the code and the framework recommended by google to develop a cross-platform mobile flutter application.
- **GitHub desktop Application** provides version control and backup functionality for effective project management. It allows you to track changes in your code, create backups, and manage different versions of your project.

- **Get:** popular state management library for Flutter, the cross-platform UI toolkit. It provides a wide range of features beyond state management, making it a comprehensive solution for building Flutter applications.

- **Flutter screen util:** The library includes helper classes or widgets that assist in building responsive user interfaces. This can involve dynamically adjusting widget sizes.

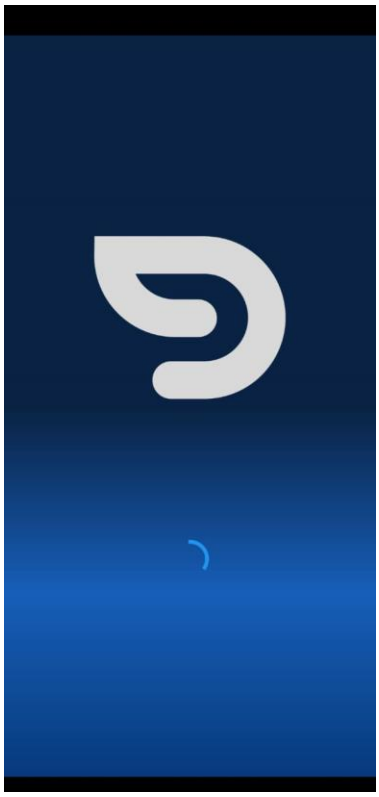
- **Flutter polyline points:** The package offers methods to calculate and draw polylines on a map. Given a list of latitude and longitude coordinates. So It can helps draw the route from source to destination

- **Geolocator :**you can access the device's location services and obtain the user's current location data. Which I use then in google maps.

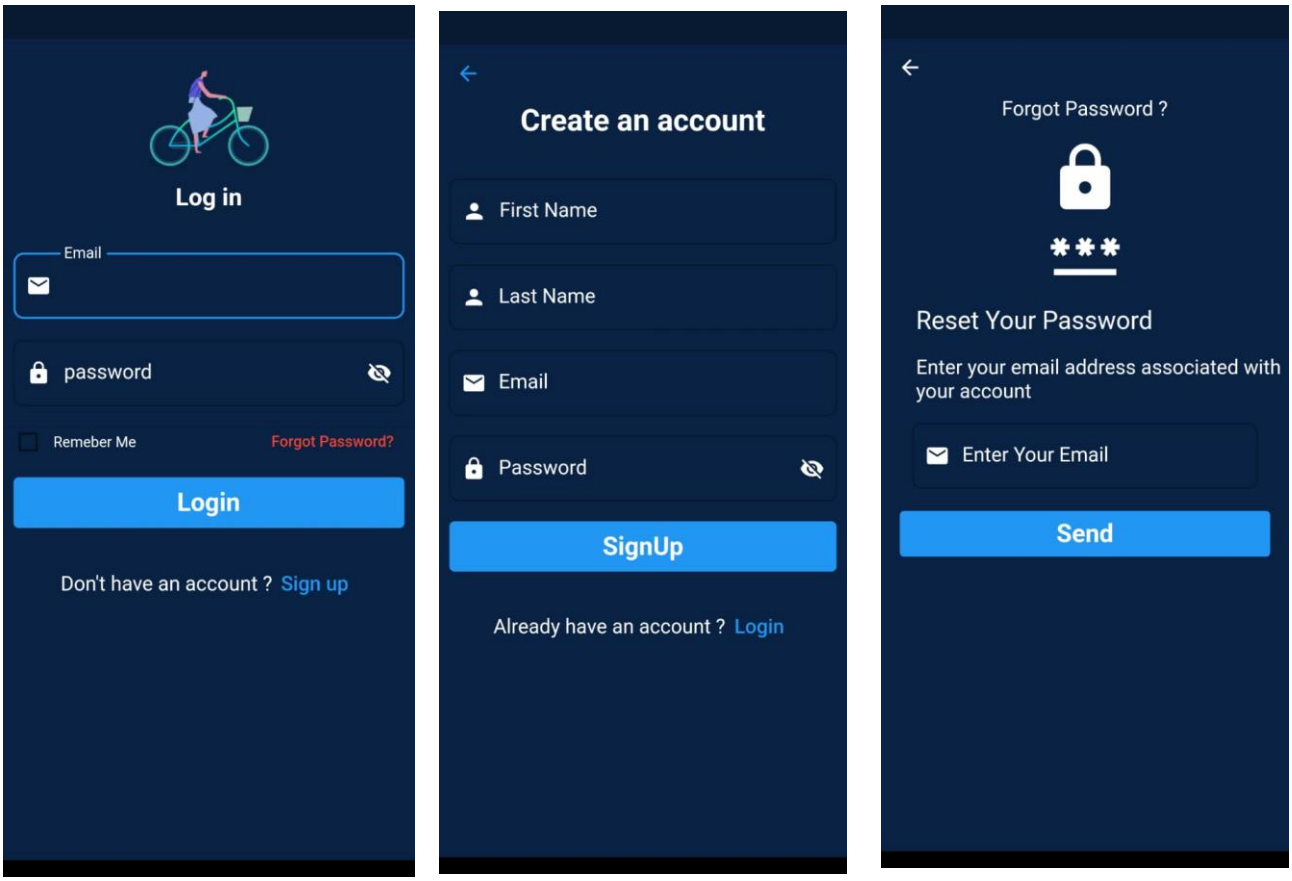
Screens and Functionalities

The app has a lot of different screens that can be used, with each having its own functionality.

The Splash screen. Which is used to check if user when he logged in choose to either stay logged in or not(remember me), if he choose to stay logged in the app take the user right straight to the home page if not go to login screen.



The Auth screens



These screen is used for authentication part with firebase and also have offline validation to make the app run faster.
they are basically contains widgets which every thing in flutter is a widget
Widget used in the auth screens are

- Text form field
- Checkbox
- Text buttons
- Show snack bar

HomeScreen& HealthScreen

which are the two main two screen after app loads , it display the distance traveled by user and body temp, smoke or not around bike , and lid flash controller which all are sensors btw.
Figure 4.3.1

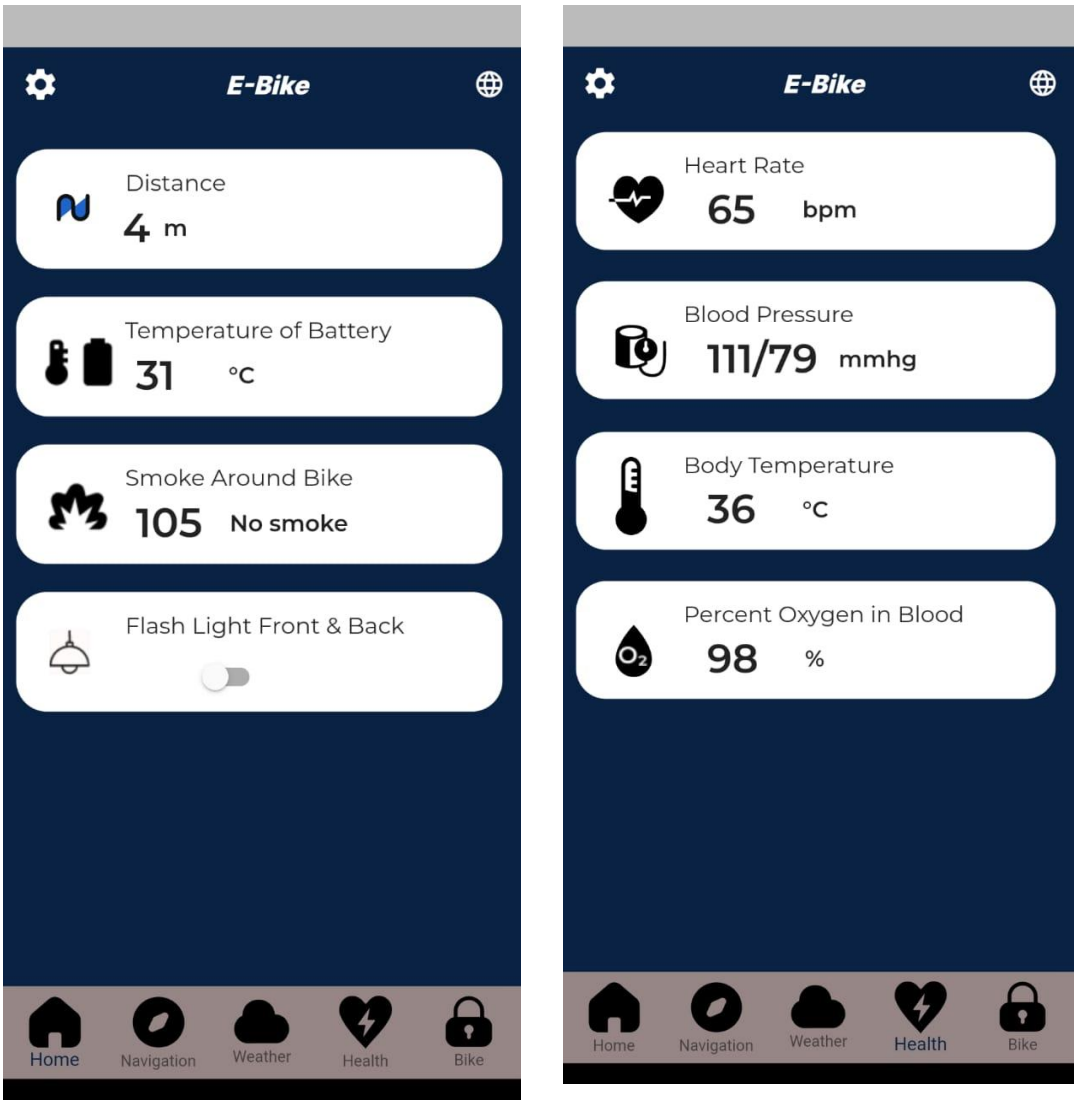


Figure 4.3.2

The health screen shows heart rate, blood pressure, blood temperature and percent ox in blood and all of this data are coming from the real time data base server.

4.4: Back-End Development

Backend (Firebase)

Authentication

I used firebase authentication for our app using email and password

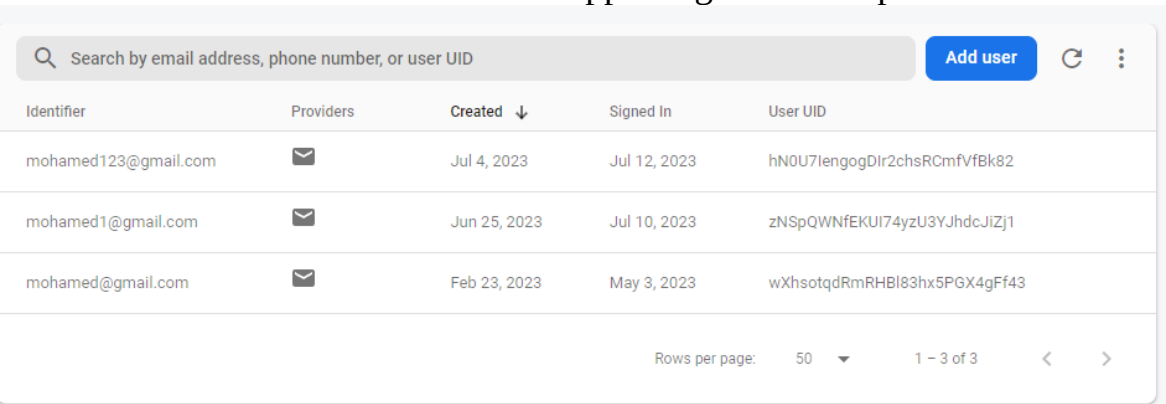


Figure 4.4.1

This code for login screen

```
Future<void> loginUser(BuildContext context) async {
  try {
    final user = await _auth.signInWithEmailAndPassword(
        email: EmailController.text, password: PasswordController.text);
    if (user != null) {
      // lets save user with shared prefrences

      SharedPreferences prefs = await SharedPreferences.getInstance();
      prefs.setString("userID", user.user!.uid);
      print(user.user!.uid);
      Get.to(() => const Nav());
      print(_auth.currentUser!.uid);
    } else {
      print('error');
    }
  } on FirebaseAuthException catch (e) {
    if (e.code == 'user-not-found') {
      showSnackBar(context, "wrong email or password");
    } else if (e.code == 'wrong-password') {
      showSnackBar(context, "wrong email or password");
```

Figure 4.4.2

For sign up authentication

```dart
Future<void> createAccount(BuildContext context) async {
  try {
    final user = await _auth.createUserWithEmailAndPassword(
        email: Emailcontroller.text, password: Passwordcontroller.text);
    final firestore = FirebaseFirestore.instance;
    firestore.collection('users').doc(user.user!.uid).set({
      "Email": Emailcontroller.text,
      "First Name": fNamecontroller.text,
      "Last Name": lNamecontroller.text,
      "Distance": 0,
      "Balance": 0,
    });
    if (user != null) {
      Get.to(() => const Nav());
      print(_auth.currentUser!.uid);
    } else {
      print('error');
    }
  } on FirebaseAuthException catch (e) {
    if (e.code == 'email-already-in-use') {
      showSnackBar(context, "The user already exists");
    }
  } catch (e) {
    print(e);
```

Figure 4.4.3

Firebase Firestore

I used this service to store users data and maintaining the reward system which is wallet screen

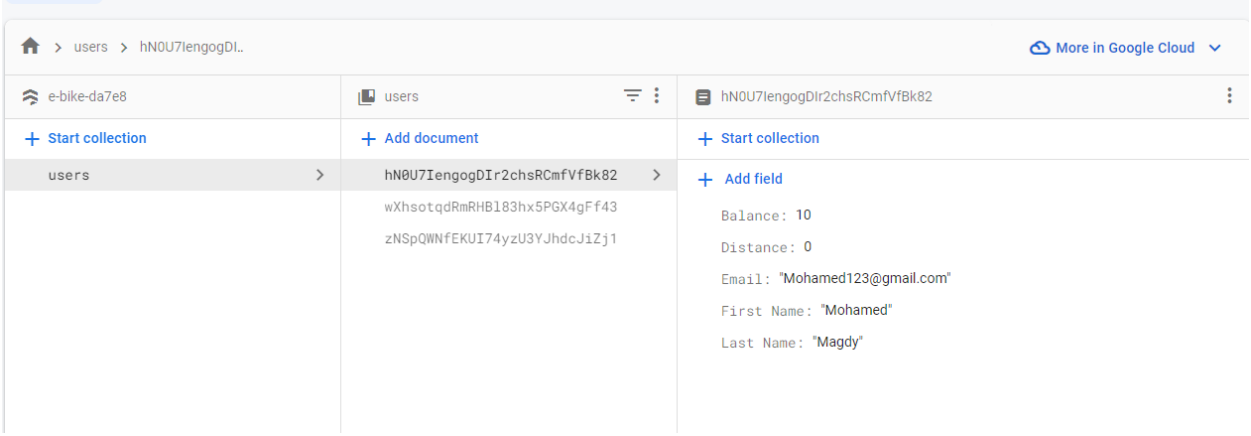


Figure 4.4.4

Here is the code for storing data of each user when signing up

```
Future<void> createAccount(BuildContext context) async {
  try {
    final user = await _auth.createUserWithEmailAndPassword(
        email: Emailcontroller.text, password: Passwordcontroller.text);
    final firestore = FirebaseFirestore.instance;
    firestore.collection('users').doc(user.user!.uid).set({
      "Email": Emailcontroller.text,
      "First Name": fNamecontroller.text,
      "Last Name": lNamecontroller.text,
      "Distance": 0,
      "Balance": 0,
    });
    if (user != null) {
```

Figure 4.4.5

In our profile screen we can read from the firestore database like so

```
  void loadUserData() {
    firestore
        .collection("users")
        .doc(_auth.currentUser!.uid)
        .get()
        .then((snapshot) {
      String firstName = snapshot.data()!['First Name'];
      String lastName = snapshot.data()!['Last Name'];
      name = '$firstName $lastName';
      email = snapshot.data()!['Email'];
      update(); // Notify GetBuilder that the data has changed
    });
  }
```

Figure 4.4.6

Real-time database

I used real time data base for managing sensor reading between the esp32 and the fire base so I can use fire base like a middle man to get the date and show it in our app

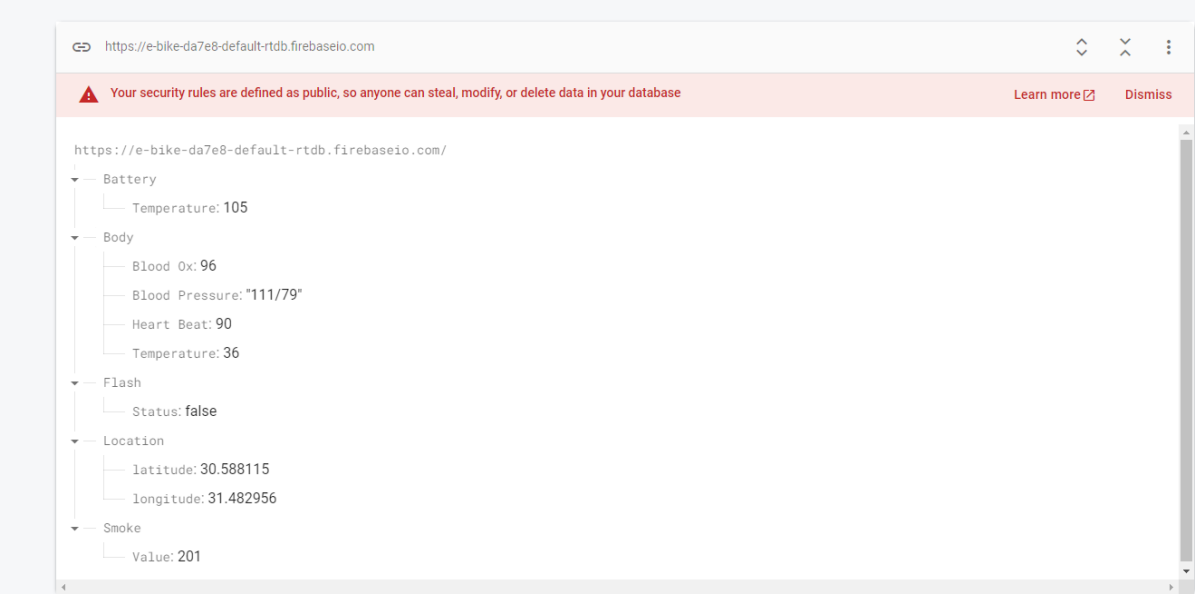


Figure 4.4.7

Here in home Screen I can get the reading from the Realtime data base through this code
using Obx or getx as stream to get updated automatically when change in the database

```
void tamperatureChange() {
  _dbref.child('Battery').child('Temperature').onValue.listen((event) {
    final data = event.snapshot.value;
    print('Temperature: $data');
    newTemp.value = int.parse(data.toString());
    update();
  });
}

void smokeChange() {
  _dbref.child('Smoke').child('Value').onValue.listen((event) {
    final data = event.snapshot.value;
    print('Value: $data');
    smoke.value = int.parse(data.toString());
    update();
  });
}
```

Figure 4.4.8

But here I can write into the real time data to change the state of the lid if I want it to be on or off

```
void updateValue() {
  _dbref.child("Flash").update({"Status": !isOn});
  update();
}
```

Figure 4.4.9

Basically, here it change the value of the path(flash/status) in real time data base from on to off or likewise
Then for the esp32 side it can use get Bool to get the status of the lid and make the lid on and that's how we can control the lid of the in front and the back of the bike with our application.

Esp32

Here is how we can connect to Wi-Fi with the esp32

```
#include <Arduino.h>
#include <Firebase_ESP_Client.h>
#define WIFI_SSID "WE"
#define WIFI_PASSWORD "0123456789"
#define MAX_BRIGHTNESS 255

#include "addons/TokenHelper.h"
#include "addons/RTDBHelper.h"

#define API_KEY "YOUR_API_KEY"

#define DATABASE_URL "YOUR_DATABASE_URL"
// Insert RTDB URLefine the RTDB URL */


// Define Firebase Data object.
FirebaseData fbdo;

// Define firebase authentication.
FirebaseAuth auth;

// Definee firebase configuration.
FirebaseConfig config;

//======================================== Mi
unsigned long sendDataPrevMillis = 0;
const long sendDataIntervalMillis = 1000; //-
//========================================

// Boolean variable for sign in status.
bool signupOK = false;
```

```
WiFi.mode(WIFI_STA);
WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
Serial.println("---------------Connection");
Serial.print("Connecting to : ");
Serial.println(WIFI_SSID);
while (WiFi.status() != WL_CONNECTED){
  Serial.print("."); delay(100);
}
Serial.println();
Serial.print("Successfully connected to : ");
Serial.println(WIFI_SSID);
//Serial.print("IP : ");
//Serial.println(WiFi.localIP());
Serial.println("---------------");

config.api_key = API_KEY;

// Assign the RTDB URL (required).
config.database_url = DATABASE_URL;
// Sign up.
Serial.println();
Serial.println("---------------Sign up");
Serial.print("Sign up new user... ");
if (Firebase.signUp(&config, &auth, "", "")){
  Serial.println("ok");
  signupOK = true;
}
else{
  Serial.printf("%s\n", config.signer.signupError.message.c_str());
}
Serial.println("---------------");

// Assign the callback function for the long running token generation
config.token_status_callback = tokenStatusCallback; //--> see addons/

Firebase.begin(&config, &auth);
Firebase.reconnectWiFi(true);


erial.println("Initializing...");
```

Figure 4.4.10

Writing inside the realtime data

```
    if (Firebase.RTDB.setInt(&fbdo, "Body/Blood Ox",spo2)) {
  Serial.println( ); Serial.println("SpO2 =");Serial.println(spo2); ;
  Serial. print(" - successfully saved to :" +fbdo.dataPath());
    Serial.println( " (" + fbdo.dataType() + ")");

  }
  else {
    Serial.println("FAILED");
    Serial.println("REASON: " + fbdo.errorReason());
  }


  // Temperature body using MAX3010
```

Figure 4.4.11

Store the spo2 value of the ox sensor into the path(body/blood)

Read from real time-database

```
  if (Firebase.RTDB.getBool(&fbdo, "Flash/Status")) {
  if(fbdo.dataType() == "boolean"){
    Status = fbdo.boolData();
    if(Status==true){
      digitalWrite(ledPin, HIGH);
      Serial.println("Light turned on");
    }else{
      // If lightVal is less than the initial reading within a th
      if (lightVal  - lightInit < 500)


digitalWrite(ledPin, LOW); // Turn off the light
        Serial.println("Light turned off");
turn on light
} else if (lightVal > 1800) {
        digitalWrite(ledPin, HIGH); // Turn on the light
        Serial.println("Light turned on");
      }


  }
```

Figure 4.4.12

Get the status of the ldr which may be on or off .

Chapter 5: Website

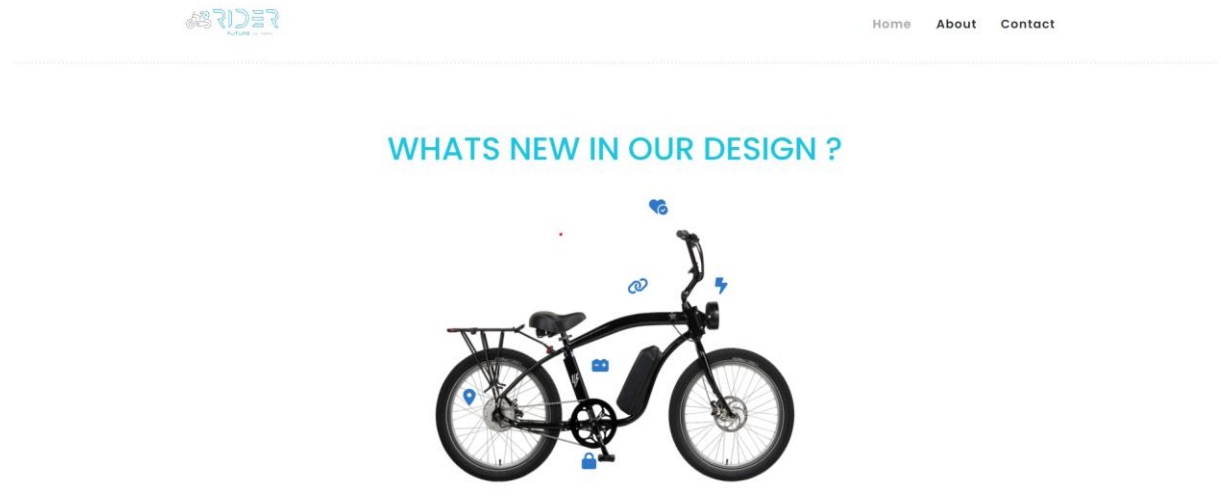
5.1: UI/UX Design & Front-End Development

In this part I will explain tools which I used to build the front-end of our small e-commerce, so let's see.

- We used Pure HTML, CSS and JavaScript to Build This website.
- We have main three pages and one take places when the visitor makes an action in addition to the pages of ordering.
- And Some Additional tools like font Awesome.
- To Link CSS file to our body we used Link tag (<Link><Link/>)
- To Link JavaScript file to our body we used Link tag (<script><script/>)

Home Page:

First Section as a **Ui**:



First Section as a **Code**:

Figure 5.1.1

1-NavBar

```
<!-- ***** Header Area Start ***** -->
<header class="header-area header-sticky">
  <div class="container">
    <div class="row">
      <div class="col-12">
        <nav class="main-nav">
          <!-- ***** Logo Start ***** -->
          <a href="index.html" class="logo">
            <img
             src="./assets/images/logo-removebg-preview.png"
              class="logo"
            />
          </a>
          <!-- ***** Logo End ***** -->
          <!-- ***** Menu Start ***** -->
          <ul class="nav">
            <li class="scroll-to-section">
              <a href="index.html" class="active">Home</a>
            </li>
            <li class="scroll-to-section">
              <a href="about.html">About</a>
            </li>
            <li class="scroll-to-section">
              <a href="contact.html">Contact</a>
            </li>
          </ul>

          <!-- ***** Menu End ***** -->
        </nav>
      </div>
    </div>
  </div>
</header>
<!-- ***** Header Area End ***** -->
```

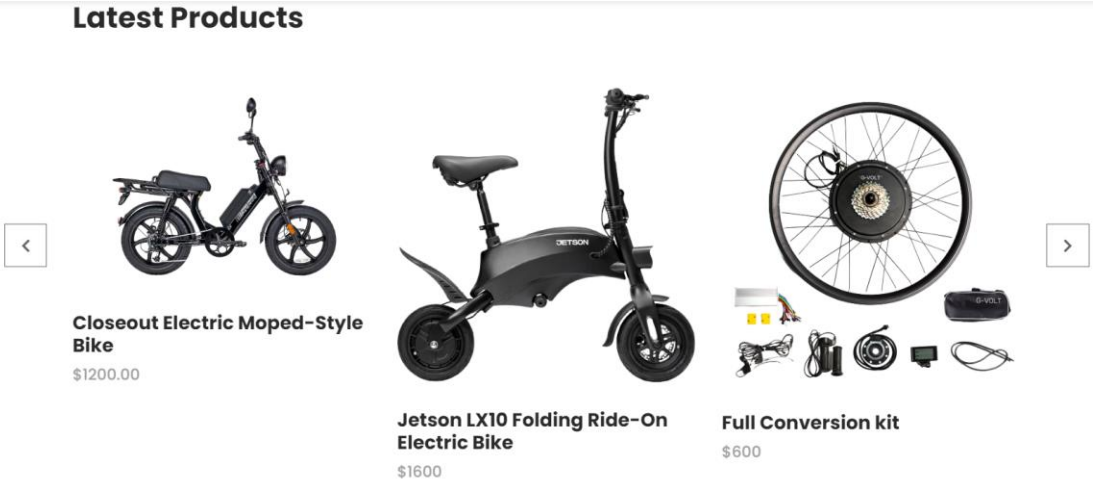
Figure 5.1.2

2-Section

```
section class="section" id="men" style="margin-top: 100px">
 <div class="feature">
   <h1>whats new in our design ?</h1>
   <div class="description">
     <img src="./assets/images/Description.png" />
     <button class="gps">
       <i
         class="fa-solid fa-location-dot fa-beat fa-lg"
         style="color: #2f78c6"
       ></i>
     </button>
     <button class="heartsensor">
       <i
         class="fa-solid fa-heart-circle-check fa-beat fa-lg"
         style="color: #2f78c6"
       ></i>
     </button>
     <button class="pedallock">
       <i
         class="fa-solid fa-lock fa-beat fa-lg"
         style="color: #2f78c6"
       ></i>
     </button>
     <button class="kit">
       <i
         class="fa-sharp fa-solid fa-car-battery fa-beat fa-lg"
         style="color: #2f78c6"
       ></i>
     </button>
     <button class="iot">
       <i
         class="fa-solid fa-link fa-beat fa-lg"
         style="color: #2f78c6"
       ></i>
     </button>
     <button class="flashlight">
<button class="flashlight">
   <i
     class="fa-solid fa-bolt-lightning fa-beat fa-lg"
     style="color: #2f78c6"
   ></i>
</button>
/div>
```

Figure 5.1.3

Products Section as a **Ui**:



Products Section as a **Code**:

```
<div class="container">
  <div class="row">
    <div class="col-lg-6">
      <div class="section-heading">
        <h2>Latest Products</h2>
      </div>
    </div>
  </div>
</div>
<div class="container">
  <div class="row">
    <div class="col-lg-12">
      <div class="men-item-carousel">
        <div class="owl-men-item owl-carousel">
          <div class="item">
            <div class="thumb">
              <div class="hover-content">
                <ul>
                  <li>
                    <a href="single-product.html"
                      ><i class="fa fa-eye"></i
                      ></a>
                  </li>
                </ul>
              </div>
              <img src="assets/Store//Closeout HyperScorpion Electric Moped-Style Bike.png" alt=''/>
            </div>
            <div class="down-content">
              <h4>Closeout Electric Moped-Style Bike</h4>
              <span>$1200.00</span>
            </div>
          </div>
```

Figure 5.1.4

```
                <div class="item">
                  <div class="thumb">
                    <div class="hover-content">
                      <ul>
                        <li>
                          <a href="single-product.html"
                            ><i class="fa fa-eye"></i
                          ></a>
                        </li>
                      </ul>
                    </div>
                    <img src="assets/Store/Full Conversion kit.png" alt="" />
                  </div>
                  <div class="down-content">
                    <h4>Full Conversion kit</h4>
                    <span>$600</span>
                  </div>
                </div>
                <div class="item">
                  <div class="thumb">
                    <div class="hover-content">
                      <ul>
                        <li>
                          <a href="single-product.html"
                            ><i class="fa fa-eye"></i
                          ></a>
                        </li>
                      </ul>
                    </div>
                    <img src="assets/Store/hub motor.webp" alt="" />
                  </div>
                  <div class="down-content">
                    <h4>Hub motor</h4>
                    <span>$400</span>
                  </div>
                </div>
              </div>
            </div>
          </div>
        </div>
      </section>
```

Footer as **Ui**:

Figure 5.1.5

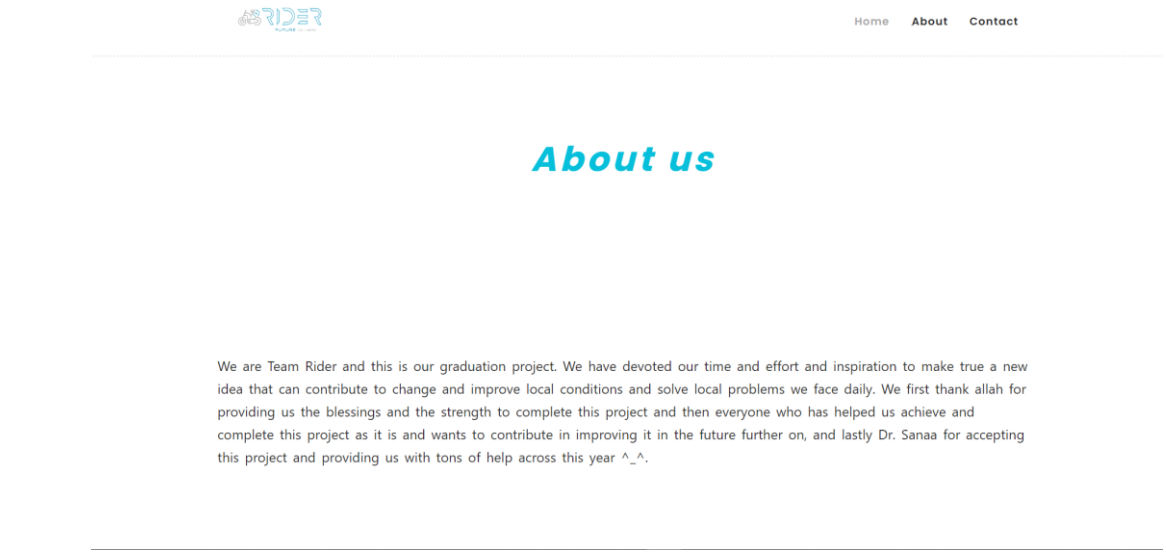
Footer as **Code**:

```
<!-- ***** Footer Start ***** -->
<footer>
  <div class="container">
    <div class="row">
      <div class="col-lg-12">
        <div class="under-footer">
          <p>Copyright © 2023</p>

          <ul>
            <li>
              <a href="#"><i class="fa fa-facebook"></i></a>
            </li>
            <li>
              <a href="#"><i class="fa fa-twitter"></i></a>
            </li>
            <li>
              <a href="#"><i class="fa fa-linkedin"></i></a>
            </li>
            <li>
              <a href="#"><i class="fa fa-behance"></i></a>
            </li>
          </ul>
        </div>
      </div>
    </div>
  </div>
</footer>
```

Figure 5.1.6

About Page as a Ui**:**



About Page as Code:
-The Same Code for Footer and Navbar

```
<!-- ***** Main Banner Area Start ***** -->
<div class="page-heading about-page-heading" id="top">
    <div class="container">
        <div class="row">
            <div class="col-lg-12">
                <div class="inner-content">
                    <h2>About us</h2>
                </div>
            </div>
        </div>
    </div>
</div>
<!-- ***** Main Banner Area End ***** -->

<!-- ***** About Area Starts ***** -->
<div class="about-us">
    <div class="container">
        <div class="row">
            <p class="pio">We are Team Rider and this is our graduation project. We have devoted our time and effort and inspiration to make true a new idea that can contribute to c
                We first thank allah for providing us the blessings and the strength to complete this project and  then everyone who has helped us achieve and complete this project
                </p>
        </div>
    </div>
</div>
<!-- ***** About Area Ends ***** -->
```

Figure 5.1.7

Contact Page as a Ui:
-The Same Code for Footer and Navbar

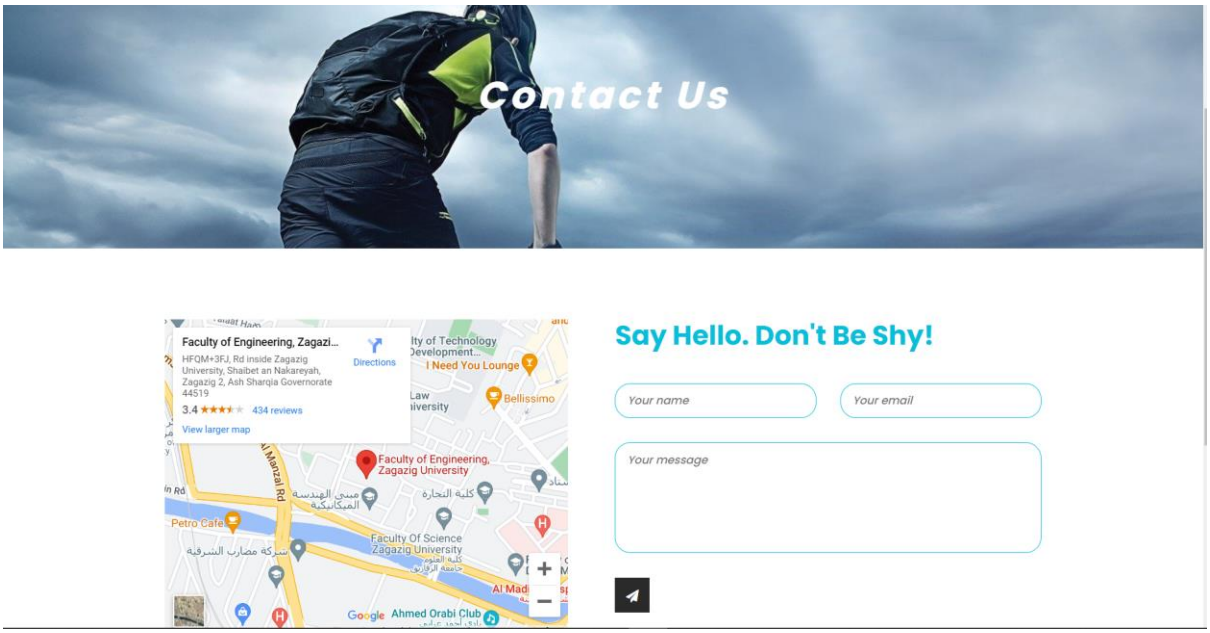


Figure 5.1.8

Contact Page as **Code**:

Figure 5.1.9

```
<!-- ***** Main Banner Area Start ***** -->
<div class="page-heading contact-page-heading" id="top">
    <div class="container">
        <div class="row">
            <div class="col-lg-12">
                <div class="inner-content">
                    <h2>Contact Us</h2>
                </div>
            </div>
        </div>
    </div>
</div>
<!-- ***** Main Banner Area End ***** -->

<!-- ***** Contact Area Starts ***** -->
<div class="contact-us">
    <div class="container">
        <div class="row">
            <div class="col-lg-6">
                <div id="map">
                    <iframe src="https://www.google.com/maps/embed?pb=!1m18!1m12!1m3!1d2648.4911689210961d31.4799805143163!3d30.
                    5869152882906371m3!1f0!2f0!3f0!3m2!1i1024!2i768!4f13.
                    1!3m3!1m2!1s0x14f7f053ae7dd993%3A0x82a66871a800f213!2sFaculty%20of%20Engineering%2C%20Zagazig%20University!5e0!3m2!1sen!2seg!4v1688730841422!5m2!1sen!2seg"
                    width="100%" height="400" style="border:0;" allowfullscreen="" loading="lazy" referrerpolicy="no-referrer-when-downgrade"></iframe>                    <!--
                    You can simply copy and paste "Embed a map" code from Google Maps for any location. -->
                </div>
            </div>
            <div class="col-lg-6">
                <div class="section-heading">
                    <h2>Say Hello. Don't Be Shy!</h2>
                </div>
                <form id="contact" action="" method="post">
                    <div class="row">
                        <div class="col-lg-6">
                            <fieldset>
                                <input name="name" type="text" id="name" placeholder="Your name" required="">
                            </fieldset>
                        </div>
                        <div class="col-lg-6">
                            <fieldset>
                                <input name="email" type="text" id="email" placeholder="Your email" required="">
                            </fieldset>
                        </div>
                        <div class="col-lg-12">
                            <fieldset>
                                <textarea name="message" rows="6" id="message" placeholder="Your message" required=""></textarea>
                            </fieldset>
                        </div>
                        <div class="col-lg-12">
                            <fieldset>
                                <button type="submit" id="form-submit" class="main-dark-button"><i class="fa fa-paper-plane"></i></button>
                            </fieldset>
                        </div>
                    </div>
                </form>
            </div>
        </div>
    </div>
</div>
<!-- ***** Contact Area Ends ***** -->
```

Figure 5.1.9

5.2: Back-End Development

In back-end part there is some important files that controls all functions in the website, There is a three main controllers
And these controllers are:
1 – ProductController
2 – OrderController
3 – MyFatoorahController

---

1 – ProductController :

This code represents a PHP class named **ProductController** within the **App\Http\Controllers\Admin** namespace. It extends the **Controller** class provided by Laravel. It handles CRUD operations for the **Product** model.
Methods:

1. **index()**: This method displays a paginated listing of products. It queries the **Product** model and passes the result to the **admin.products.index** view.
2. **create()**: This method displays a form for creating a new product. It returns the **admin.products.create** view.
3. **store(Request $request)**: This method stores a newly created product in the database. It validates the incoming request data, creates a new **Product** instance, and saves it to the database. If images are included in the request, it associates them with the product. It uses database transactions to ensure data consistency. If successful, it redirects to the **products.index** route with a success message. Otherwise, it rolls back the transaction and redirects back with an error message.
4. **edit(string $id)**: This method displays a form for editing a specific product identified by its **$id** parameter. It fetches the product from the database and passes it to the **admin.products.edit** view.
5. **update(Request $request, string $id)**: This method updates the specified product in the database. It validates the incoming request data, finds the product by its **$id**, and updates its attributes. If images are included in the request, it removes the existing images associated with the product and associates the new images. It uses database transactions to ensure data consistency. If successful, it redirects to the **products.index** route with a success message. Otherwise, it rolls back the transaction and redirects back with an error message.

6. **destroy(string $id)**: This method deletes the specified product from the database. It finds the product by its **$id**, deletes it, and redirects to the **products.index** route with a success message.

---

2 – OrderController :

This code represents a PHP class named **OrderController** within the **App\Http\Controllers\Admin** namespace. It extends the **Controller** class provided by Laravel. It handles the display and printing of orders.

Methods:

1. **index()**: This method displays a paginated listing of orders. It queries the **Order** model and retrieves a paginated collection of orders. The collection is then passed to the **admin.orders.index** view.
2. **show(int $id)**: This method displays the details of a specific order. It fetches the order from the database using the given **$id** parameter. If the order is found, it is passed to the **admin.orders.show** view for rendering. If the order is not found, a 404 error is returned.
3. **print(int $id)**: This method displays a printable version of a specific order. It fetches the order from the database using the given **$id** parameter. If the order is found, it is passed to the **admin.orders.print** view for rendering. If the order is not found, a 404 error is returned.

---

3 – MyFatoorahController :

This code represents a PHP class named **MyFatoorahController** that extends the **Controller** class. It handles the integration with the MyFatoorah payment gateway for creating invoices, processing callbacks, and updating payment statuses.

Properties:

- **$mfObj**: An instance of the **PaymentMyfatoorahApiV2** class from the MyFatoorah library.

Methods:

1. **__construct()**: The constructor method initializes the **PaymentMyfatoorahApiV2** object. It takes the API key, country ISO, and test mode configurations from the Laravel configuration file.
2. **index($user, $order)**: This method creates a MyFatoorah invoice for a specific user and order. It sets the payment method ID (0 for MyFatoorah invoice) and retrieves the invoice URL using the **getInvoiceURL()** method

of the **$mfObj** object. If successful, it redirects the user to the invoice URL. If an exception occurs, it redirects with an error message.

3. **getPayLoadData($user, $order)**: This private method prepares the payload data required to create a MyFatoorah invoice. It retrieves the user's name, order price, email, and other information to pass as parameters to the **getInvoiceURL()** method.
4. **callback()**: This method processes the callback from MyFatoorah after payment. It retrieves the payment ID from the request and uses the **getPaymentStatus()** method of the **$mfObj** object to check the payment status. Depending on the payment status (Paid, Failed, or Expired), it updates the order's payment status and redirects the user to the appropriate route with success or error messages.

5.3: Database

Database Overview

Overview

Our project includes a database implemented using MySQL that hosts the backend data of the system. This includes the data of the administration part of the products as well as the orders and users registrations.

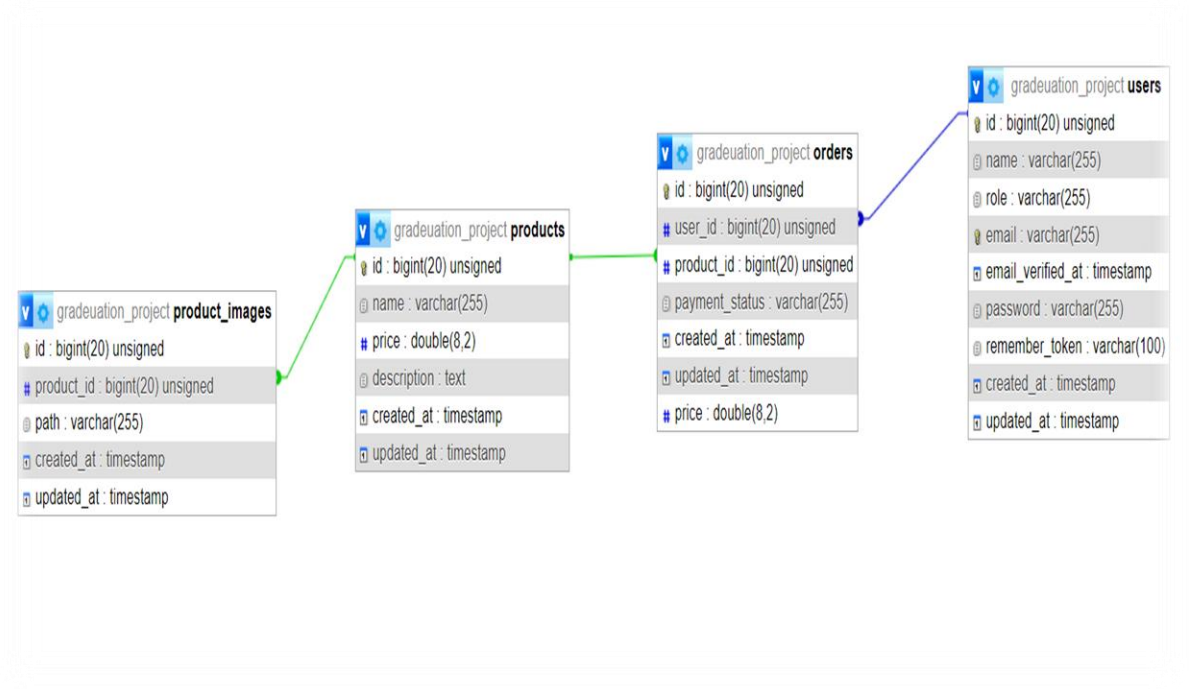


Figure 5.3.1

Technical Overview

The entity relationship diagram of the database consists of the following main tables:

1. Users: it is a table for user who are allowed to buy an electric bike or part of it, which is the same admin's table that allows him to market our products, add modifications continuously, and give permissions to users to order the products that they want on the web page.
2. Orders: This table includes the orders of the users by which products are selected and bought.
3. Products: A table that includes the electric bike and its parts , which was displayed on the web page for sale to users.

Users' Table :

```
163
164 CREATE TABLE `users` (
165   `id` bigint UNSIGNED NOT NULL,
166   `name` varchar(255) COLLATE utf8mb4_unicode_ci NOT NULL,
167   `role` varchar(255) COLLATE utf8mb4_unicode_ci NOT NULL DEFAULT 'user',
168   `email` varchar(255) COLLATE utf8mb4_unicode_ci NOT NULL,
169   `email_verified_at` timestamp NULL DEFAULT NULL,
170   `password` varchar(255) COLLATE utf8mb4_unicode_ci NOT NULL,
171   `remember_token` varchar(100) COLLATE utf8mb4_unicode_ci DEFAULT NULL,
172   `created_at` timestamp NULL DEFAULT NULL,
173   `updated_at` timestamp NULL DEFAULT NULL
174 ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;
```

Figure 5.3.2

Users's table, which includes the most important attribute (role attribute) to distinguish between the user and the admin,and when the admin requests to enter the dashboard page, he must enter his email and password, and when purchasing the user from our products on the web page, he is asked to write the name, email and password in order to be stored this data is in the database.

```
88 CREATE TABLE `orders` (
89   `id` bigint UNSIGNED NOT NULL,
90   `user_id` bigint UNSIGNED NOT NULL,
91   `product_id` bigint UNSIGNED NOT NULL,
92   `payment_status` varchar(255) COLLATE utf8mb4_unicode_ci NOT NULL DEFAULT 'pending',
93   `created_at` timestamp NULL DEFAULT NULL,
94   `updated_at` timestamp NULL DEFAULT NULL,
95   `price` double(8,2) NOT NULL
96 ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;
97
```

Figure 5.3.3

Orders' Table :

Orders's table links the user with the products he purchased, and includes the total price of the order and the payment status determined by the user.

Products' Table :

```
135 CREATE TABLE `products` (
136   `id` bigint UNSIGNED NOT NULL,
137   `name` varchar(255) COLLATE utf8mb4_unicode_ci NOT NULL,
138   `price` double(8,2) NOT NULL,
139   `description` text COLLATE utf8mb4_unicode_ci NOT NULL,
140   `created_at` timestamp NULL DEFAULT NULL,
141   `updated_at` timestamp NULL DEFAULT NULL
142 ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;
143
```

Figure 5.3.4

Product's table It includes the names, descriptions and prices of the electric bike parts that were displayed on the web page.

Product_images' Table :

```
150 CREATE TABLE `product_images` (
151   `id` bigint UNSIGNED NOT NULL,
152   `product_id` bigint UNSIGNED NOT NULL,
153   `path` varchar(255) COLLATE utf8mb4_unicode_ci NOT NULL,
154   `created_at` timestamp NULL DEFAULT NULL,
155   `updated_at` timestamp NULL DEFAULT NULL
156 ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;
```

Figure 5.3.5

Finally, we made a table linked to the products's table in order to allow it to display more than one image of a single product.

Future Improvements: -

one of the main problems we faced an could be resolved with more intimate research and development is full integration of the IoT sensor network with the motor controller into one overall controller and connecting certain features on the main separate controller to work on mobile application via wifi/Bluetooth

Another improvement is to further enhance battery efficiency and range vs physical size formula of the battery pack as well as faster charging or integration with electric power deparment's projects like fast charging stations and fast high power chargers.

Also improvement of sensor network efficiency and response as well as measurement accuracy of sensors and compatibility as we had issues with compatibility between certain sensors before.

Lastly, local manufacturing of BLDC hub motors and their controllers using local materials and factories will further boost this idea's viability as a successful business and a pioneer one on the E-bike market in Egypt.

Conclusion: -

In conclusion, the development of an electric bike (e-bike) for this graduation project has been a fascinating and rewarding experience. The project has involved extensive research, design, and testing, resulting in the creation of a fully functional e-bike prototype that meets the project objectives.

The e-bike prototype has several notable features, including a powerful electric motor, a long-lasting battery, and a user-friendly interface. These features make the e-bike a viable alternative to traditional bicycles and gas-powered vehicles for commuting and recreational purposes.

The project has also highlighted the potential of e-bikes to reduce carbon emissions and promote sustainable transportation. As the world faces increasing concerns about climate change and environmental sustainability, e-bikes can play an essential role in reducing carbon emissions and promoting sustainable transportation.

References: