



Project Report on **IOT Based Electrical Vehicle Platform**

By

MEHUL MAGGO 19104083

AASHISH SINGH BISHT 19104091

ADHYAN MAHAJAN 19104094

GAUTAM KUMAR 19104109

HRIDYANSH MEENA 19104111

GURSHAAN SINGH 19104114

Under supervision of

Prof. DHIRAJ BHARAT

Department of Electrical Engineering,

Pec, Chandigarh

December 2022

DECLARATION

We hereby declare that our project work entitled “IOT Based Electrical Vehicle Platform” is an authentic record of our own work as a requirement of this Major Project under the able guidance of Prof Dhiraj Bharat.

Date : (19th December 2022)

(Group 3) :

MEHUL MAGGO 19104083

AASHISH SINGH BISHT 19104091

ADHYAN MAHAJAN 19104094

GAUTAM KUMAR 19104109

HRIDYANSH MEENA 19104111

GURSHAAN SINGH 19104114

Approval by Mentor

Certified that the above statement made by the students is correct to the best of our knowledge and belief.

Prof Dhiraj Bharat

Electrical Engineering Department

Punjab Engineering College

ACKNOWLEDGEMENT

We have taken efforts in this project. However, it would not have been possible without the kind support and help of many individuals and organizations. I would like to extend my sincere thanks to all of them.

We would like to thank our mentor Prof. Dhiraj Bharat for guiding us throughout the process of making this Major Project.

We would like to extend our gratitude to authors of the papers and information sources without which this project would not have been possible.

We would also sincerely like to thank our institution, Punjab Engineering College, Chandigarh for this wonderful opportunity which will act as a big milestone in our career development and I will strive to use skills and knowledge acquired during the tenure in best possible way.

INDEX

S.No.	Title	Page No.
1	Abstract	5
2	Introduction	6
3	Research Methodology	7
4	Blockchain Technology	8
5	Electrical Hardware	11
6	Circuit Coding	17
7	Mobile Application	20
8	Load Curve Forecasting Using Machine Learning	24
9	Future Work	29
10	Conclusion	30
11	Result	31

ABSTRACT

The proliferation of electric vehicles would significantly affect the power distribution infrastructure if appropriate measures are not implemented about the high-power consumption needed for charging EVs. Therefore, a platform must be created to assist charging point operators in successfully managing EV users' charging requests and ensuring that their needs are satisfied without consuming an excessive amount of the distribution grid's capacity. If this is not done, EV owners will have a problem in a few years with the scarcity of charging ports and grid congestion brought on by the simultaneous charging of all EVs.

A smart EV charging infrastructure built on the blockchain is suggested. The charging demand (kWh) and maximum length of the charging event provided by the EV user are utilized to identify and utilize the EV load flexibility through smart charging to provide a stable grid. The platform connects EV owners with charging stations, reducing the number of participants in the EV charging ecosystem. Flexibility (power and time) in EV charging is sold on the blockchain network. As a result, the acceptance rate of EVs may greatly increase if this architecture is employed.

INTRODUCTION

If proper precautions are not taken about the high-power consumption required for charging EV, the rise of electric vehicles will have a significant impact on the power distribution grid. As a result, a platform must be developed to help charging point operators manage EV users' charging demands effectively and guarantee that their needs are met without using up too much of the distribution grid's capacity. The lack of charging outlets and grid congestion brought on by the simultaneous charging of all EVs will be an issue for EV owners in a few years if this is not done.

EV smart charging is still in its infancy. Nearly twenty times as much energy as an average North American home is needed to charge an EV. Despite this high charging power, grid congestion caused by EV charging is currently infrequent. This is due to the low penetration rate of EVs in many nations and the installation of charge points (CP) concurrently with substation reinforcement.

Smart electric vehicle (EV) charging uses intelligence to manage when and how an electric vehicle plugged into a smart charger will receive power for charging based on the cost of electricity, its availability, and the needs of the driver. You can regulate, monitor, and modify energy consumption with EV smart charging. It needs a data link between the EV, the charger, the grid, and the cloud-based charging management platform of the charge point operator.

In this report, a blockchain-based smart EV charging infrastructure is proposed. The EV load flexibility is identified and used through smart charging to produce a stable grid using the charging demand (kWh) and maximum duration of the charging event provided by the EV user. Through the platform, EV owners and charging stations are connected, minimising the number of players in the EV charging ecosystem. Within the blockchain platform, flexibility (power and time) in charging of EVs is traded. As a result, if this architecture is used, the acceptance rate of EVs may rise significantly.

RESEARCH METHODOLOGY

Software Used:

1. Arduino
2. Jupyter Notebook
3. Flutter
4. Solidity

The steps taken for the project, follows the process:

1. Arduino is used to link a hardware circuit to a wifi module and utilise the wifi module to transfer data from the circuit's load to a mobile application.
2. The machine learning programme that helped us forecast the load curve was created in a Jupyter notebook.
3. The mobile application that allows us to manage the supply of the load as well as the completion of the IOT system that allows us to obtain data on power consumption by the load were both developed using the Flutter SDK.
4. The smart contracts for our blockchain platform, which will manage the charge request, were created using Solidity. Additionally, it was used to establish the Ethereum wallet for interactions between the EV user and the corporation that operates the charging station.

BLOCKCHAIN TECHNOLOGY

Blockchain is a decentralised, immutable database that makes it easier to track assets and record transactions in a corporate network. An asset may be physical (such as a home, car, money, or land) or intangible (intellectual property, patents, copyrights, branding). On a blockchain network, practically anything of value may be recorded and traded, lowering risk and increasing efficiency for all parties.

Importance of Blockchain

Information is essential to business. It is best if it is received quickly and is accurate. Blockchain is the best technology for delivering that information because it offers real-time, shareable, and entirely transparent data that is kept on an immutable ledger and accessible exclusively to members of a permissioned network. Among other things, a blockchain network can track orders, payments, accounts, and production. Additionally, because everyone has access to the same version of the truth, you can see every aspect of a transaction from beginning to end, increasing your confidence and opening up new prospects.

Key elements of Blockchain

Distributed Ledger Technology: The distributed ledger and its immutable record of transactions are available to all network users. Transactions are only recorded once with this shared ledger, preventing the duplication of effort present in conventional corporate networks.

Immutable Records: Once a transaction has been added to the shared ledger, no participant is permitted to alter or interfere with it. A fresh transaction must be added to undo an error in a transaction record before both transactions are displayed.

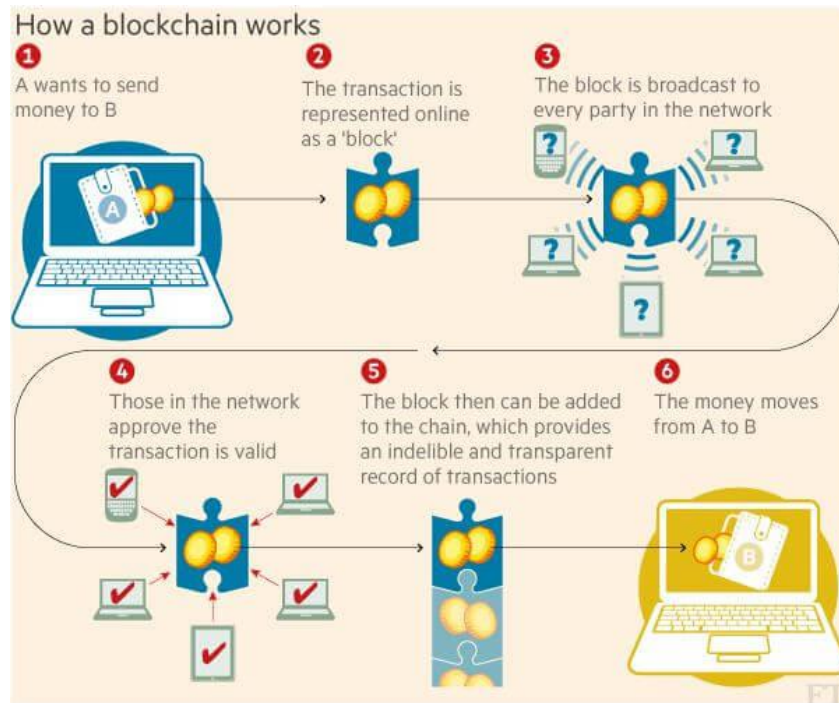
Smart Contracts: A set of instructions known as a smart contract is saved on the blockchain and automatically carried out to speed up transactions. A smart contract can specify parameters for corporate bond transfers, stipulate how much must be paid for travel insurance, and much more.

How blockchain works?

Blockchain operates through a multi-step process, which can be summarised as follows:

1. An authorised participant enters a transaction, which the technology must authenticate.
2. A block that reflects that particular transaction or piece of data is produced by that action.
3. Every computer node in the network receives the block.
4. A block is added to the current blockchain after the transaction has been verified by authorised nodes. (Miners are nodes in public blockchain networks; they are frequently compensated for this labour using a procedure known as Proof of Work, or PoW; typically, they receive payment in the form of bitcoin.)

5. The transaction is completed when the update is delivered throughout the network.



BC technology is built around four key ideas:

1. **Peer-to-peer networks:** By removing the central TTP and indicating that each network node has the same rights, this solution eliminates the central TTP. Nodes in this network can communicate with one another by using a set of private and public keys. The public key serves as an address on the network, and the private key is used to sign transactions.
2. **Open and distributed ledger:** Consider a ledger as an accounting book that records all of the network's transactions in time order. The fact that each node has a copy of this data structure means that it is not a centralised entity. The ledger is available to everyone and is public. The location of the asset and the amount that each user has in their account are both visible to everyone on the network. Additionally, every node in the network has the ability to determine if a transaction is valid or invalid.
3. **Ledger copies synchronization:** A method to synchronise ledgers across nodes is required in this type of scenario, where nodes each have their own copy of the same ledger. Three primary steps must be taken in order to achieve this goal: (a) broadcasting the new transactions openly to the network; (b) validating the new transactions; and (c) adding the validated transactions to the ledgers.
4. **Mining:** Because of network latency, not every node in a distributed system receives transactions—or, more specifically, blocks of transactions—at the same time. The chain must only have a legal and ordered branch, hence it is necessary to stop every node from adding transactions to it.

Blockchain in Internet of Things:

The Internet of Things (IoT) is a pervasive network of people and intelligent physical things, sometimes known as "Things." IoT turns the physical world into a massive information system by enabling any "Thing" to connect and interact. Cloud computing, machine learning, data analysis, and information modelling are just a few of the technologies that are increasingly becoming a crucial component of the IoT fabric.

As IoT items proliferate and become more visible on the Internet, security—or, more specifically, ensuring that only authorised users have access to resources—becomes increasingly important. On the one hand, IoT's pervasiveness fosters the development of creative applications for the end user, but on the other hand, a lack of security measures may result in serious problems like people being subjected to physical harm like burglary owing to a smart alarm system being hacked.

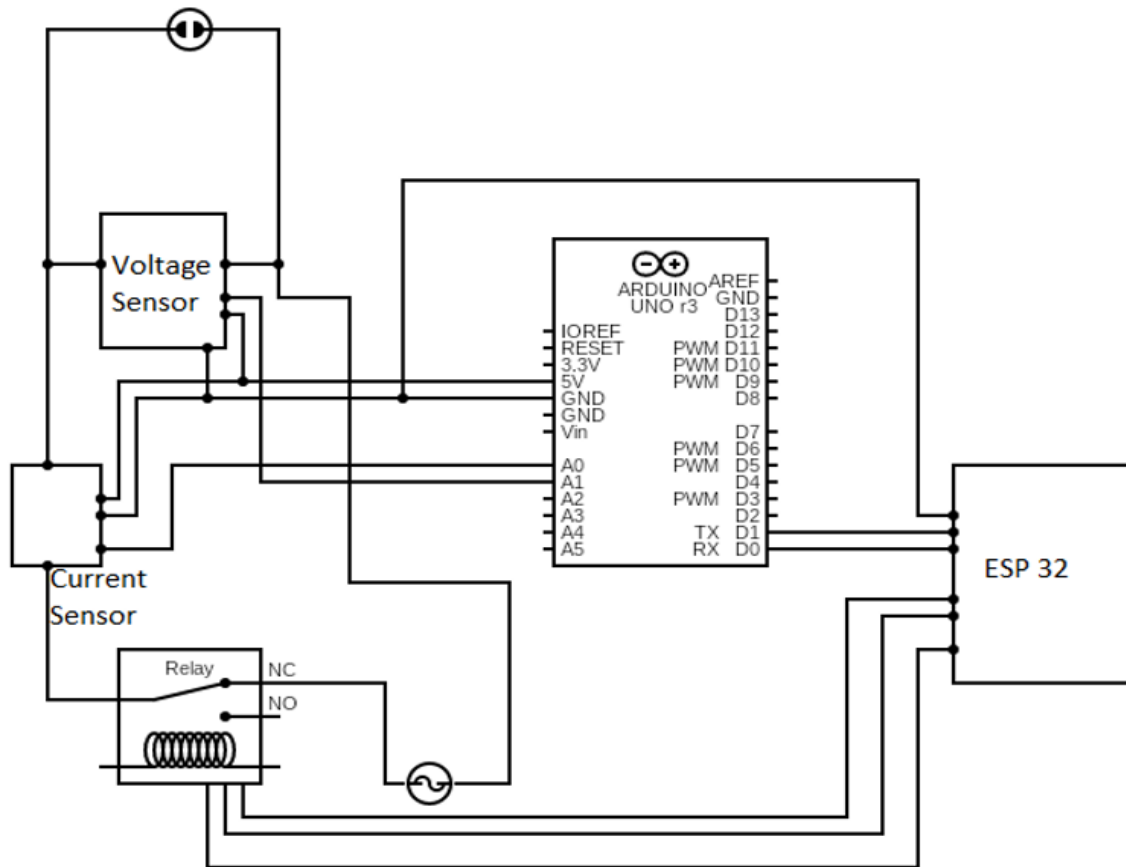
For the development of such IoT settings, a distributed trust solution that ensures scalability, privacy, and stability is a key component. Due to its inherent security, the Blockchain (BC) technology has advanced greatly in recent years and is now seen as a promising solution in attaining the aforementioned objectives. Due to its features, including immutability, transparency, auditability, data encryption, and operational resilience, BC is actually a "safe by design" system that can reduce security concerns.

Crypto Wallet:

A cryptocurrency wallet enables users to store, send, and receive cryptocurrency, much like any other digital wallet. It is a piece of software that saves cryptocurrencies securely and maintains a record of all transactions (buying, selling, and lending). A cryptocurrency wallet can be readily downloaded and installed on a smartphone or any other compatible device by users. It's just like any other software or a wallet that you use for your day-to-day transactions. Highly secure i.e., It is just a matter of securing your private key. Allows instant transactions across geographies. And these are barrier-free, without intermediaries. Low transaction fees i.e., The cost of transferring funds is much lower than with traditional banks. Allows transactions across multiple cryptocurrencies. This helps you do easy currency conversions.

ELECTRICAL HARDWARE

Circuit Diagram:



Components Of Circuit:

Arduino

Arduino is an open-source electronics platform based on easy-to-use hardware and software. Arduino boards are able to read inputs - light on a sensor, a finger on a button, or a Twitter message - and turn it into an output - activating a motor, turning on an LED, publishing something online.



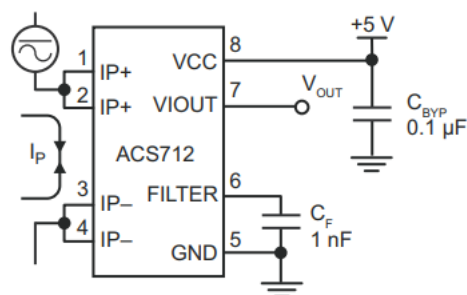
Specification Sheet for Arduino Uno is below:

Microcontroller	ATmega328P
Operating Voltage	5V
Input Voltage Recommended	7-12V
Digital I/O Pins	16
PWM Pins	6 (Pin 3, 5, 6, 9, 10 and 11)
Analog Inputs	6
Communication Protocol	UART, SPI, I2C
DC Current Per I/O Pin	20mA
DC Current for 3.3V Pin	50Ma
ICSP header	2
Flash memory	32 KB
SRAM	2 KB
EEPROM	1 KB
Length	68.6 mm
Width	53.4 mm
Weight	25g

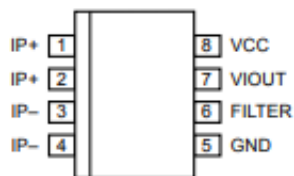
Arduino board take Analog inputs from voltage and current sensors and sends the data to Wi-fi module. Digital Pin is used for actuating the relay for switching operation.

ACS712 Current Sensor

The Allegro® ACS712 provides economical and precise solutions for AC or DC current sensing in industrial, commercial, and communications systems. The device package allows for easy implementation by the customer. Typical applications include motor control, load detection and management, switched-mode power supplies, and overcurrent fault protection.



Pin-out Diagram

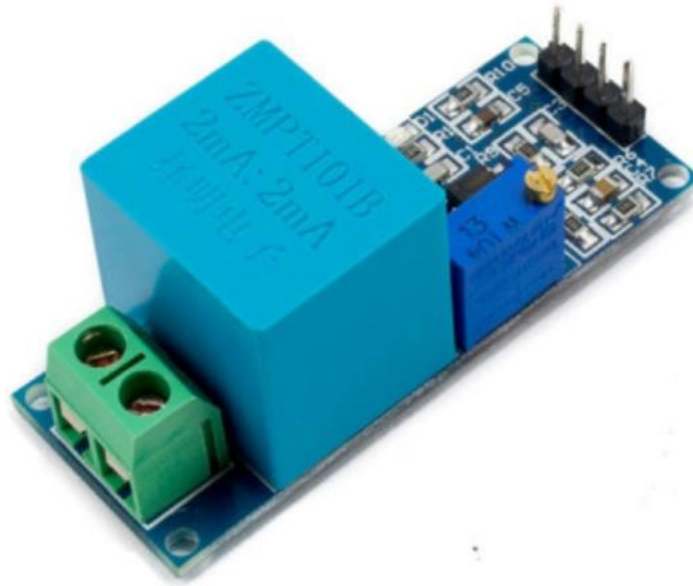


Terminal List Table

Number	Name	Description
1 and 2	IP+	Terminals for current being sensed; fused internally
3 and 4	IP-	Terminals for current being sensed; fused internally
5	GND	Signal ground terminal
6	FILTER	Terminal for external capacitor that sets bandwidth
7	VIOUT	Analog output signal
8	VCC	Device power supply terminal

Voltage Sensor – ZMPT101B

ZMPT101B AC Single Phase voltage sensor module is based on a high precision ZMPT101B voltage Transformer. ZMPT101B AC Voltage Sensor is the best for the purpose of the DIY project, where we need to measure the accurate AC voltage with a voltage transformer. This is an ideal choice to measure the AC voltage using Arduino/ESP8266/Raspberry Pi like an opensource platform. In many electrical projects, engineer directly deals with measurements with few basic requirements like High galvanic isolation, Wide Range, High accuracy, Good Consistency.

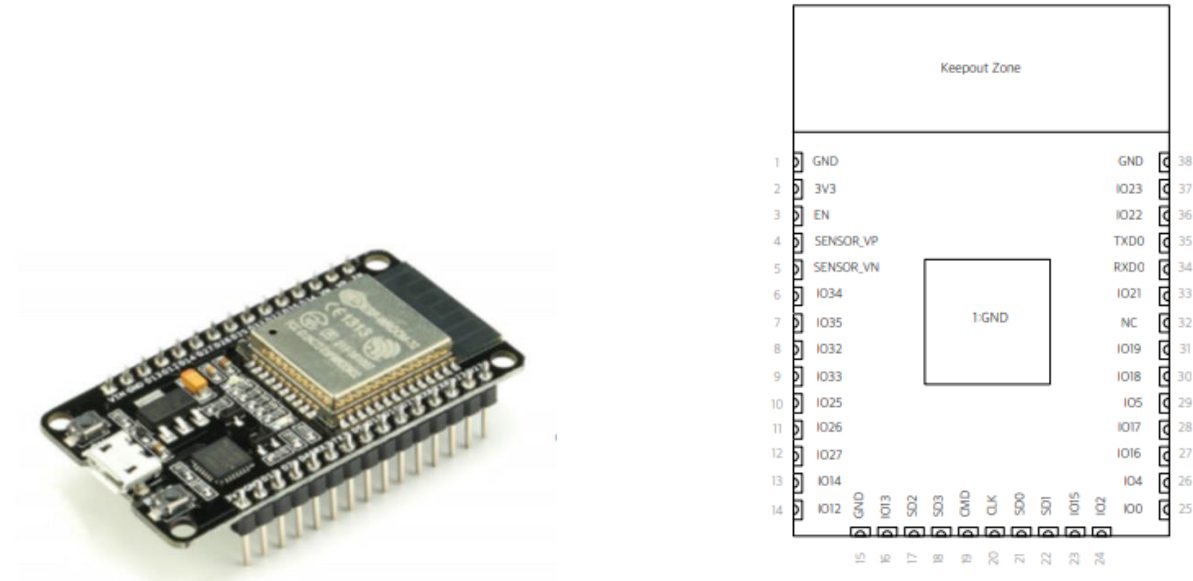


Features

- Voltage up to 250 volts can be measured
- Light weight with on-board micro-precision voltage transformer
- High precision on-board op-amp circuit
- Operating temperature: 40°C ~ + 70°C
- Supply voltage 5 volts to 30 volts

ESP32 Wi-Fi Module

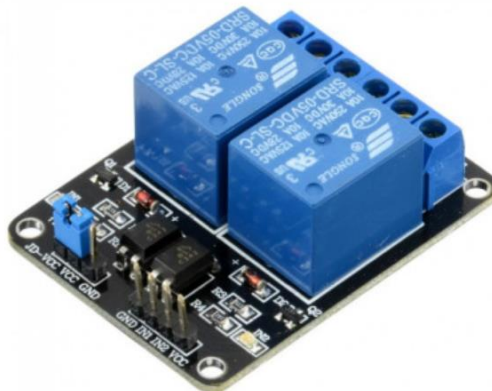
ESP32-WROOM-32 (ESP-WROOM-32) is a powerful, generic Wi-Fi + BT + BLE MCU module that targets a wide variety of applications, ranging from low-power sensor networks to the most demanding tasks, such as voice encoding, music streaming and MP3 decoding.



Above is the Pin Diagram of ESP32 wi-fi Module. It has 38 pins. The pin 35, 34 are significant in the project. They are I/O pins for transmission and receiving data from the Arduino. Wi-fi frequency range is 2.4 GHz – 2.5 GHz. Operating voltage is 5 V from the USB cable.

Relay Module

This is a 5V, 10A 2-Channel Relay interface board. It can be used to control various appliances, and other equipment's with large current.



A power relay module is an electrical switch that is operated by an electromagnet. The electromagnet is activated by a separate low-power signal from a micro controller. When activated, the electromagnet pulls to either open or close an electrical circuit.

It can be controlled directly with 3.3V or 5V logic signals from a microcontroller (Arduino, 8051, AVR, PIC, DSP).

It has a 1x4 (2.54mm pitch) pin header for connecting power (5V and 0V), and for controlling the 2 relays.



Relay module consists of six pins such as normally open pin, normally closed, common, signal, Vcc and ground pins. Signal Pin: It is used to control the relay. This pin can be active low or active high. In case of active low, the relay will activate when we apply an active low signal to the signal pin.

A digital input is given from Arduino to actuate the relay channels.

Working

Current Sensor and voltage sensor are connected in series and parallel to the load, respectively. When the load is turned ON, Current sensor and voltage sensor sends their output to A0 and A1 of Arduino, respectively. Arduino Code is burned on the microcontroller via USB connected to the PC. TX and RX pin of Arduino is connected to Rx and Tx of ESP-32 module. Esp-32 module sends data over the air to the mobile application.

Switching operation is done by the Relay module. Signal ON/OFF is sent via mobile application to ESP-32, which sends it to Arduino via Tx pin. Arduino sends digital signal (D7) to the Relay Module input for switching operation.

Load used in the circuit is a Bulb, ON/OFF operation and Energy Consumption calculation is analogous to an EV charger. Thus, display of power consumption and switching operation can be done via mobile application.

CIRCUIT CODING

Arduino IDE

The Arduino integrated development environment (IDE) is a cross-platform application (for Microsoft Windows, macOS, and Linux) that is written in the Java programming language. It originated from the IDE for the languages Processing and Wiring. It includes a code editor with features such as text cutting and pasting, searching and replacing text, automatic indenting, brace matching, and syntax highlighting, and provides simple one-click mechanisms to compile and upload programs to an Arduino board. It also contains a message area, a text console, a toolbar with buttons for common functions and a hierarchy of operation menus. The source code for the IDE is released under the GNU General Public License, version 2.

The Arduino IDE supports the languages C and C++ using special rules of code structuring. The Arduino IDE supplies a software library from the Wiring project, which provides many common input and output procedures. User-written code only requires two basic functions, for starting the sketch and the main program loop, that are compiled and linked with a program stub `main()` into an executable cyclic executive program with the GNU toolchain, also included with the IDE distribution. The Arduino IDE employs the program `avrdude` to convert the executable code into a text file in hexadecimal encoding that is loaded into the Arduino board by a loader program in the board's firmware.

From version 1.8.12, Arduino IDE windows compiler supports only Windows 7 or newer OS. On Windows Vista or older one gets "Unrecognized Win32 application" error when trying to verify/upload program. To run IDE on older machines, users can either use version 1.8.11, or copy "arduino-builder" executable from version 11 to their current install folder as it's independent from IDE.

Code for Arduino Uno and Esp32

The Arduino board and the Esp32 module in the circuit has been coded using the Arduino ide.

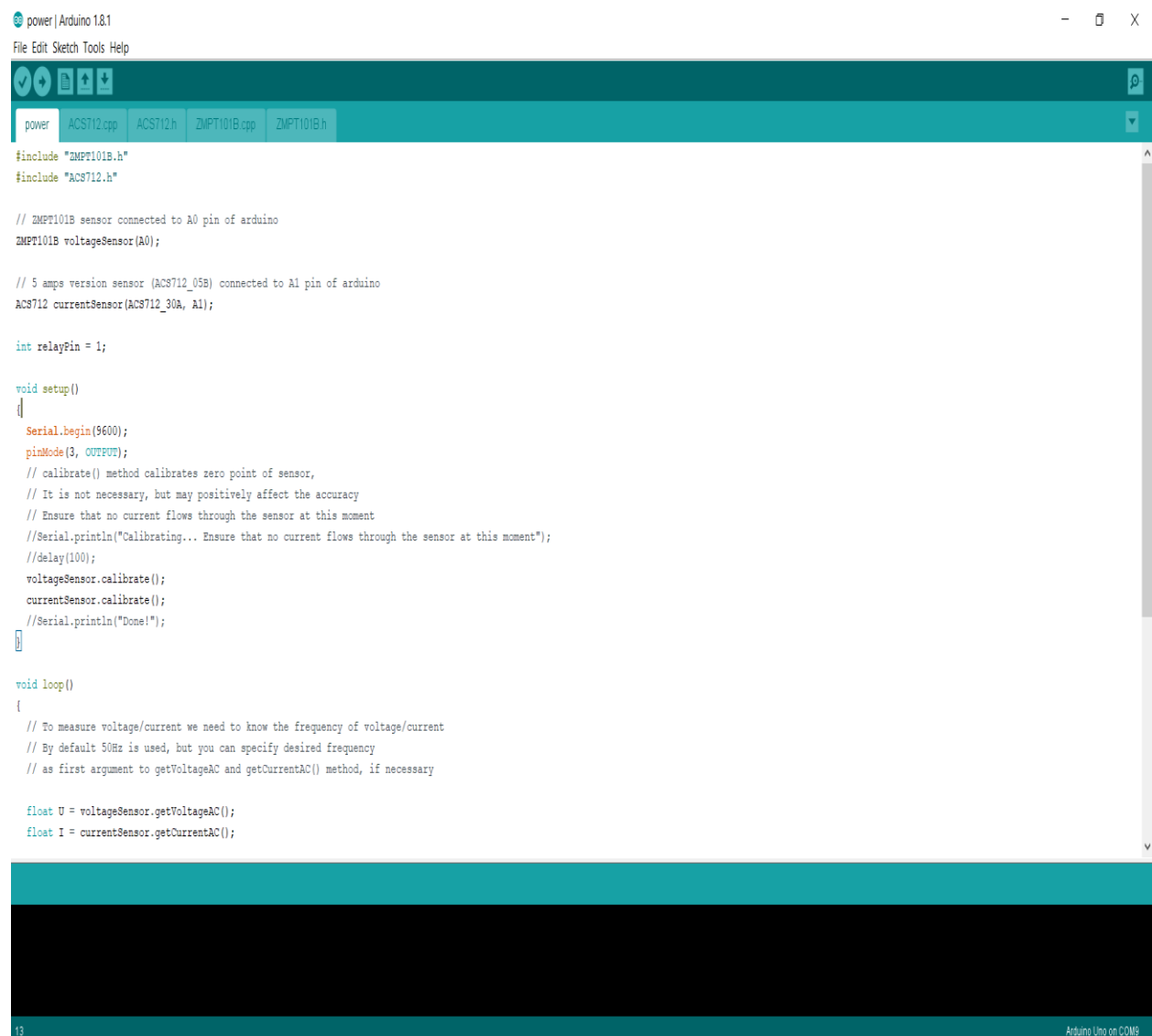
The programming language used are C and C++. First the analog output from the analog output pins A0, A1 on the Arduino Uno Board consisting of the output of the voltage and current sensor respectively is read from the Arduino board. These values are the calibrated to give us the final values of current and voltage in the proper units. The power is then calculated from the values.

The power value calculated is then sent to the Esp32 module (Bluetooth/Wifi module) using serial communication. In telecommunication and data transmission, serial communication is the process of sending data one bit at a time, sequentially, over a communication channel or computer bus. This is in contrast to parallel communication, where several bits are sent as a whole, on a link with several parallel channels.

Serial communication is used for all long-haul communication and most computer networks, where the cost of cable and synchronization difficulties make parallel communication impractical.

The Esp32 is then coded to send the power values to a mobile device using Bluetooth connection between the two. The mobile device can also send the on/off signal to the Esp32 module, the on/off signal is used to control the relay in the circuit which in turn switches the circuit on and off.

Code Snippets



```
power | Arduino 1.8.1
File Edit Sketch Tools Help

#include "ZMPT101B.h"
#include "ACS712.h"

// ZMPT101B sensor connected to A0 pin of arduino
ZMPT101B voltageSensor(A0);

// 5 amps version sensor (ACS712_05B) connected to A1 pin of arduino
ACS712 currentSensor(ACS712_30A, A1);

int relayPin = 1;

void setup()
{
  Serial.begin(9600);
  pinMode(3, OUTPUT);
  // calibrate() method calibrates zero point of sensor,
  // It is not necessary, but may positively affect the accuracy
  // Ensure that no current flows through the sensor at this moment
  //Serial.println("Calibrating... Ensure that no current flows through the sensor at this moment");
  //delay(100);
  voltageSensor.calibrate();
  currentSensor.calibrate();
  //Serial.println("Done!");
}

void loop()
{
  // To measure voltage/current we need to know the frequency of voltage/current
  // By default 50Hz is used, but you can specify desired frequency
  // as first argument to getVoltageAC and getCurrentAC() method, if necessary

  float U = voltageSensor.getVoltageAC();
  float I = currentSensor.getCurrentAC();
}
```

test_bluetooth | Arduino 1.8.1

File Edit Sketch Tools Help

test_bluetooth

```
#include <BluetoothSerial.h>
#define RXP2 16
#define TXP2 17
#define RelayPin 26 //D26
#if !defined(CONFIG_BT_ENABLED) || !defined(CONFIG_BLUEDROID_ENABLED)
#error Bluetooth is not enabled! Please run 'make menuconfig' to and enable it
#endif

//int toggleState = 1;
BluetoothSerial SerialBT;

void setup() {
  Serial.begin(115200);

  pinMode(RelayPin, OUTPUT);
  digitalWrite(RelayPin,HIGH);
  /* If no name is given, default 'ESP32' is applied */
  /* If you want to give your own name to ESP32 Bluetooth device, then */
  /* specify the name as an argument SerialBT.begin("myESP32Bluetooth"); */
  SerialBT.begin();
  Serial2.begin(9600, SERIAL_8N1, RXP2, TXP2);
  Serial.println("Bluetooth Started! Ready to pair...");
}

void loop() {
  if (Serial.available())
  {
    SerialBT.write(Serial.read());
  }
  if (SerialBT.available())
  {
    String res = SerialBT.readString();
    Serial.print(res);
    //Serial.write(SerialBT.read());
  }
}
```

Arduino Uno on COM5

MOBILE APPLICATION

Flutter

Flutter is an open-source UI software development kit created by Google. It is used to develop cross-platform applications for Android, iOS, Linux, macOS, Windows, Google Fuchsia, and the web from a single codebase.

First described in 2015, Flutter was released in May 2017.

Framework Architecture

The major components of Flutter include:

- Dart platform
- Flutter engine
- Foundation library
- Design-specific widgets
- Flutter Development Tools (DevTools)

Dart Language

Flutter apps are written in the Dart language and make use of many of the language's more advanced features.

While writing and debugging an application, Flutter runs in the Dart virtual machine, which features a just-in-time execution engine. This allows for fast compilation times as well as "hot reload", with which modifications to source files can be injected into a running application. Flutter extends this further with support for stateful hot reload, where in most cases changes to source code are reflected immediately in the running app without requiring a restart or any loss of state.

For better performance, release versions of Flutter apps on all platforms use ahead-of-time (AOT) compilation, except for on the Web where code is transpiled to JavaScript.

Flutter inherits Dart's Pub package manager and software repository, which allows users to publish and use custom packages as well as Flutter-specific plugins.

Flutter engine

Flutter's engine, written primarily in C++, provides low-level rendering support using either Google's Skia graphics library or the custom "Impeller" graphics layer. Additionally, it interfaces with platform-specific SDKs such as those provided by Android and iOS to implement accessibility, file and network I/O, native plugin support, and more.

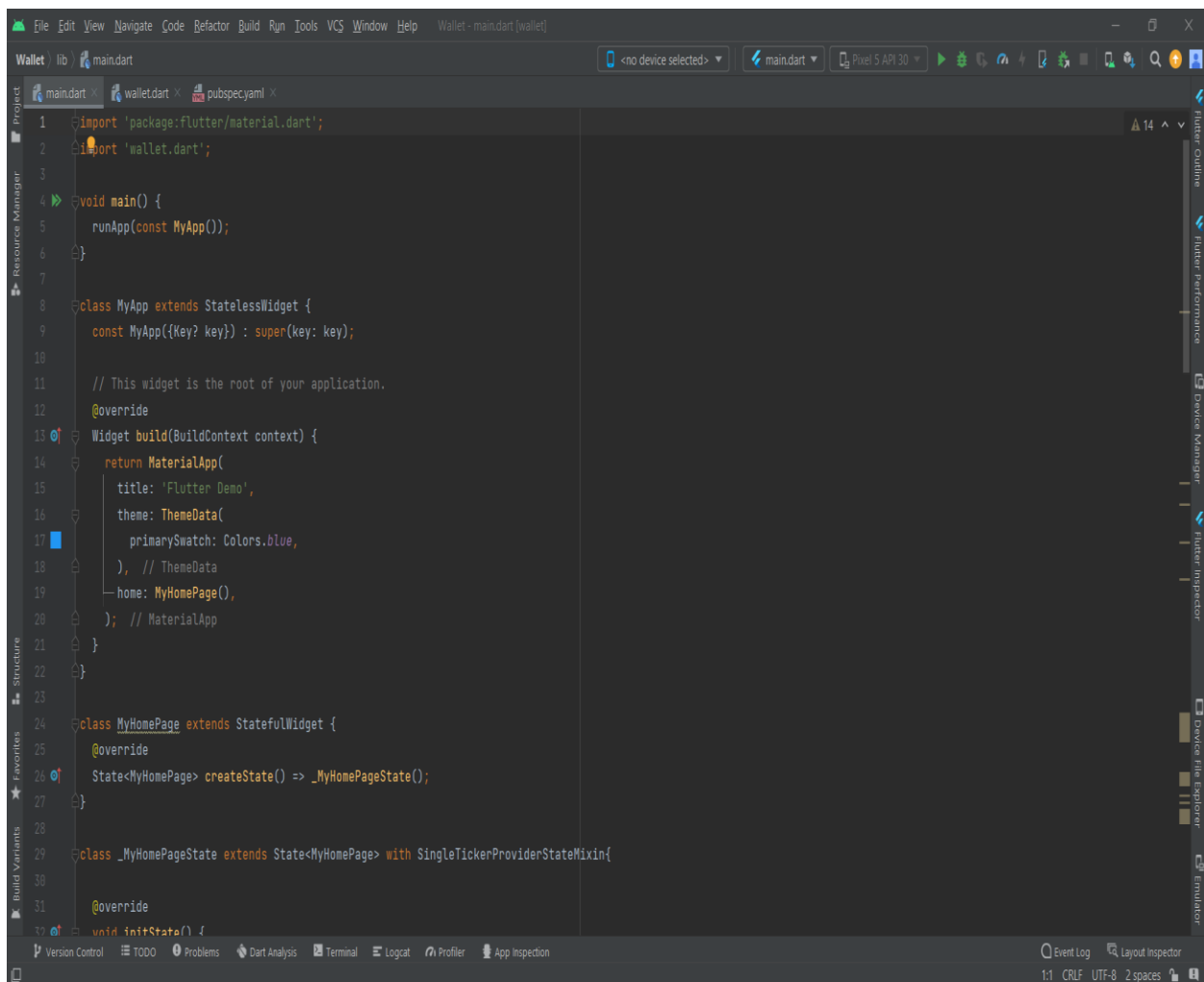
Foundation library

The Foundation library, written in Dart, provides basic classes and functions that are used to construct applications using Flutter, such as APIs to communicate with the engine.

Design-specific widgets

The Flutter framework contains two sets of widgets that conform to specific design languages: Material Design widgets implement Google's design language of the same name, and Cupertino widgets implement Apple's iOS Human interface guidelines.

App Code Snippet

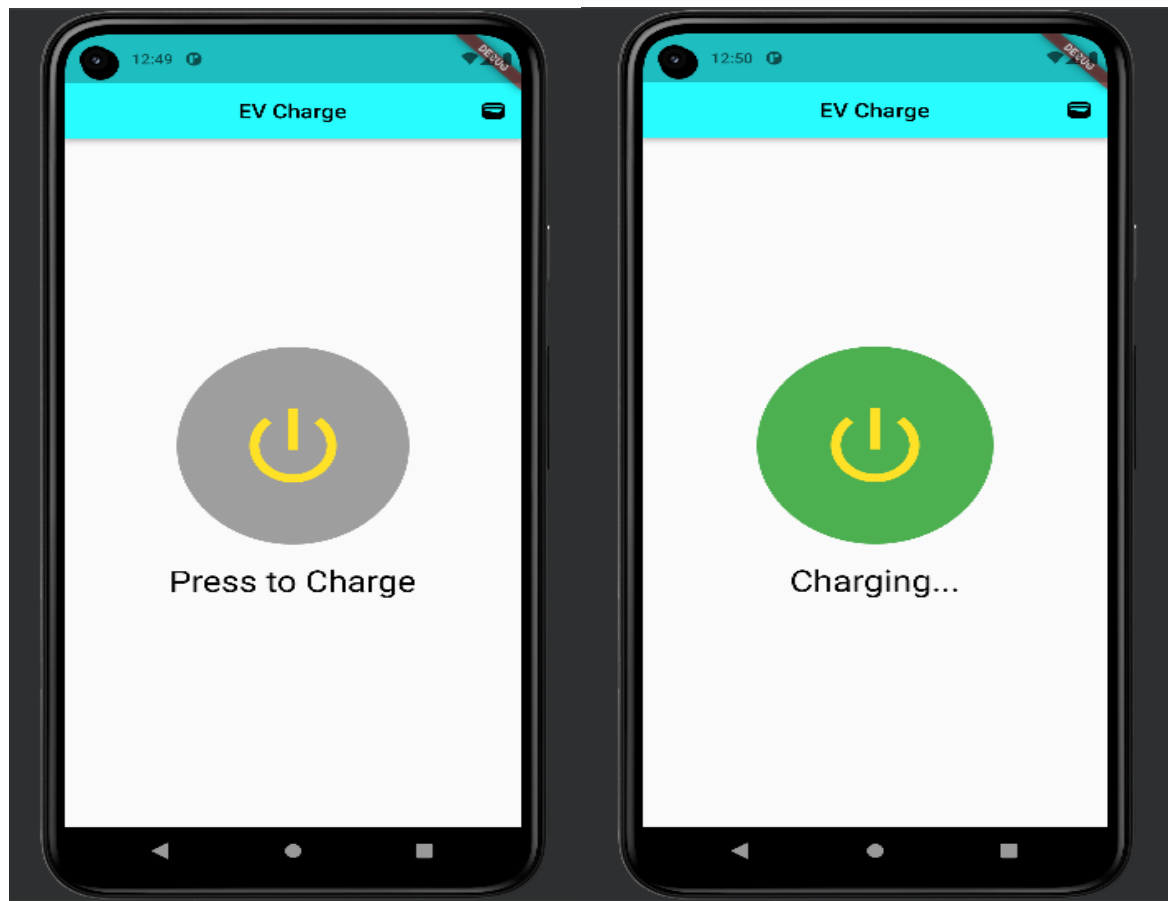


The screenshot shows an IDE window titled 'Wallet - main.dart [wallet]'. The code is as follows:

```
1 import 'package:flutter/material.dart';
2 import 'wallet.dart';
3
4 void main() {
5   runApp(const MyApp());
6 }
7
8 class MyApp extends StatelessWidget {
9   const MyApp({Key? key}) : super(key: key);
10
11   // This widget is the root of your application.
12   @override
13   Widget build(BuildContext context) {
14     return MaterialApp(
15       title: 'Flutter Demo',
16       theme: ThemeData(
17         primarySwatch: Colors.blue,
18       ), // ThemeData
19       home: MyHomePage(),
20     ); // MaterialApp
21   }
22 }
23
24 class MyHomePage extends StatefulWidget {
25   @override
26   State<MyHomePage> createState() => _MyHomePageState();
27 }
28
29 class _MyHomePageState extends State<MyHomePage> with SingleTickerProviderStateMixin {
30
31   @override
32   void initState() {
```

The IDE interface includes a sidebar with 'Project', 'Resource Manager', 'Structure', 'Favorites', and 'Build Variants'. The bottom status bar shows '1:1 CRLF UTF-8 2 spaces'.

APP UI





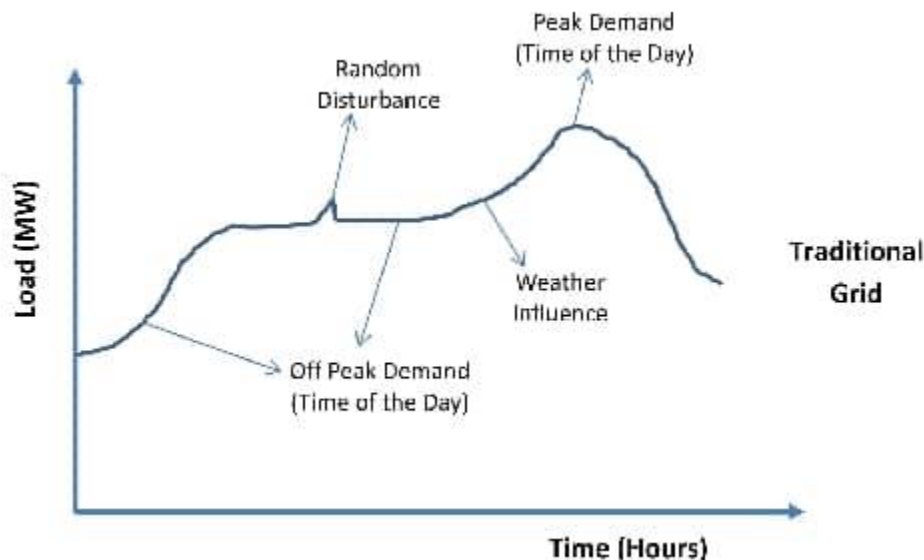
LOAD CURVE FORECASTING USING MACHINE LEARNING

Modern technology is automating traditional processes, and computers are taking control. Similarly, smart grids are taking the place of conventional energy distribution grids in order to facilitate efficient distribution and demand-side management.

Need for Load Forecasting

For the operation and planning of a utility firm, precise forecasting models for electric power loads are crucial. An electric utility can use load forecasting to make key choices, such as those regarding the generation and purchase of electricity, load shedding, and infrastructure development. For energy providers, ISOs, financial institutions, and other actors in the production, transmission, distribution, and markets of electric energy, load estimates are crucial.

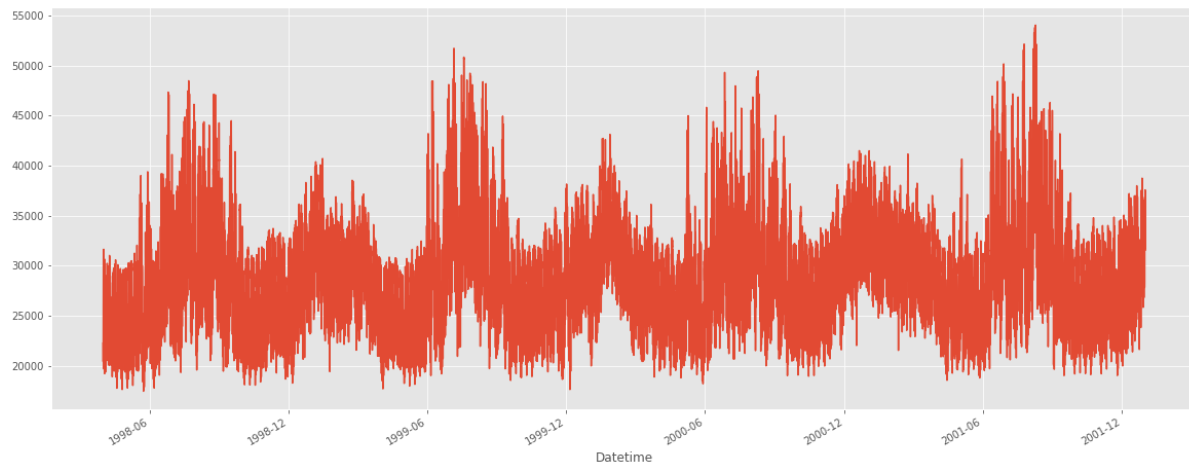
In our use case however, we require continuous load forecasting as a backend model so as to always have the current total load or stress on a grid. Based on which we will calculate the incentives given to the user for charging in non-peak time as well as imposing penalty or extra charge for charging their electric vehicles at peak time.



When it comes to load forecasting, which is a challenging multi-variable and multi-dimensional estimate problem, forecasting techniques like curve fitting with numerical approaches fall short of producing accurate results since they can't reliably follow the trends that appear to be random, a task that machine learning algorithms are better at.

Understanding the numerous factors that influence electricity demand is the first step in creating a machine learning model for load forecasting. It relies on a number of factors, including the time of day, historical patterns in electricity usage, the climate, humidity, electricity price, etc. But in order to keep things simple, we have based our forecasts on historical energy trends.

Dataset Used: The dataset in our model is in the form of time series. A time series is a sequence of numerical data points in successive order. These points are often measured at regular intervals (every month, every day, every hour, etc.). The hourly data frequency utilized in this dataset was measured from 2004 October 1 to 2018 August 3. There are 121,271 raw data items in all. This dataset is publically available and is taken from the following source: <https://www.kaggle.com/robikscube/hourly-energy-consumption>



The power consumption data points correspond to hourly load of Dayton, United States and is in megawatt (MW). Dayton is a city in western Ohio.

The code files along with the dataset can be found on this GitHub repository:

https://github.com/ApolloNow/Ethercharge_Major_Project

	A	B
1	Datetime	MW
2	31-12-2004 01:00	1596
3	31-12-2004 02:00	1517
4	31-12-2004 03:00	1486
5	31-12-2004 04:00	1469
6	31-12-2004 05:00	1472
7	31-12-2004 06:00	1518
8	31-12-2004 07:00	1598

Methodology: For the given time series, our deep learning algorithm's primary goal is to identify a function (f) such that:

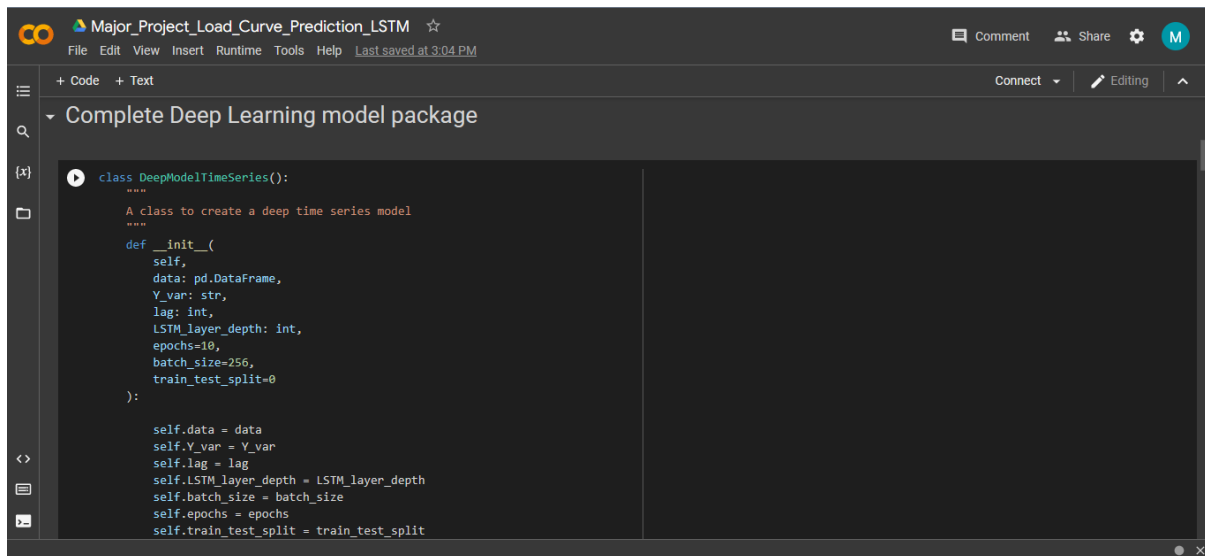
$$f(t) = f(t-1, t-2, t-3, \dots, t-n)$$

In other words, we wish to estimate a function that, given p lags of the same energy consumption, explains the present energy consumption numbers.

Algorithm Used: Recurrent neural networks are a form of artificial neural network created to recognise patterns in sequences of data, such as numerical times series data from sensors, stock

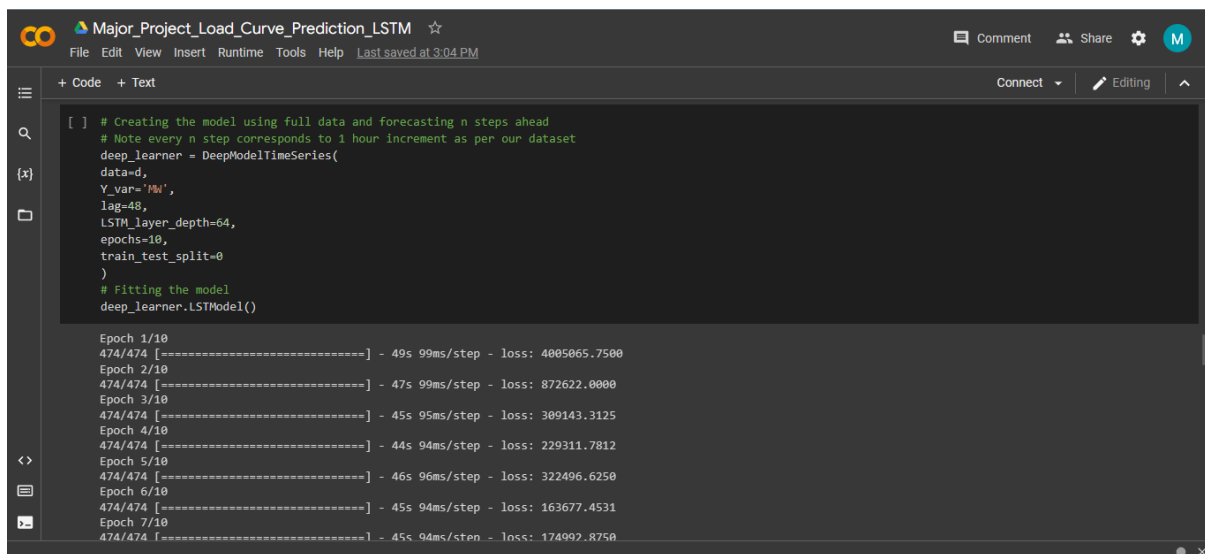
markets, and governmental organisations, of which LSTMs ("long short-term memory" units) are the most potent and well-known subset. RNNs and LSTMs vary from other neural networks in that they have a temporal dimension and take time and sequence into account. According to research, they are among the most effective and practical types of neural networks.

Implementation and Output: Screenshots taken from our Google Colab notebook.



```
class DeepModelTimeSeries():
    """
    A class to create a deep time series model
    """
    def __init__(
        self,
        data: pd.DataFrame,
        Y_var: str,
        lag: int,
        LSTM_layer_depth: int,
        epochs=10,
        batch_size=256,
        train_test_split=0
    ):

        self.data = data
        self.Y_var = Y_var
        self.lag = lag
        self.LSTM_layer_depth = LSTM_layer_depth
        self.batch_size = batch_size
        self.epochs = epochs
        self.train_test_split = train_test_split
```



```
[ ] # Creating the model using full data and forecasting n steps ahead
# Note every n step corresponds to 1 hour increment as per our dataset
deep_learner = DeepModelTimeSeries(
    data=d,
    Y_var='PM',
    lag=48,
    LSTM_layer_depth=64,
    epochs=10,
    train_test_split=0
)
# Fitting the model
deep_learner.LSTMModel()

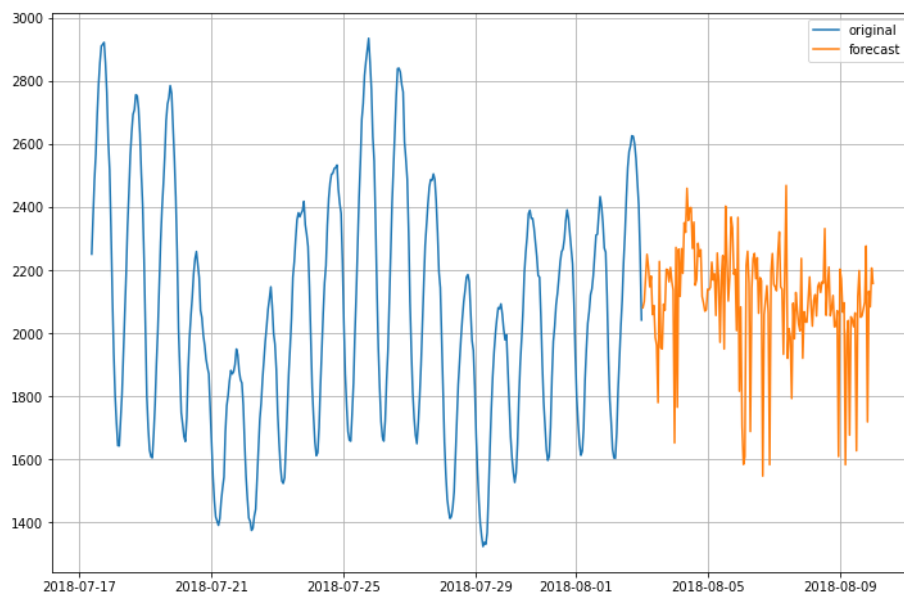
Epoch 1/10
474/474 [=====] - 49s 99ms/step - loss: 4005065.7500
Epoch 2/10
474/474 [=====] - 47s 99ms/step - loss: 872622.0000
Epoch 3/10
474/474 [=====] - 45s 95ms/step - loss: 309143.3125
Epoch 4/10
474/474 [=====] - 44s 94ms/step - loss: 229311.7812
Epoch 5/10
474/474 [=====] - 46s 96ms/step - loss: 322496.6250
Epoch 6/10
474/474 [=====] - 45s 94ms/step - loss: 163677.4531
Epoch 7/10
474/474 [=====] - 45s 94ms/step - loss: 174992.8750
```

```

Major_Project_Load_Curve_Prediction_LSTM ☆
File Edit View Insert Runtime Tools Help Last saved at 3:04 PM
+ Code + Text
Connect Editing
# Forecasting n steps ahead
# Note every n step corresponds to 1 hour increment as per our dataset
n_ahead = 168
# 168 hours corresponds to 1 week ahead
yhat = deep_learner.predict_n_ahead(n_ahead)
yhat = [y[0][0] for y in yhat]

```

Generated Load Curve:



Incentive Calculation: The following logic was implemented:

- Observing the trends from past 1 year of load demand we divide the data into 10 quantiles also known as decile analysis based on electricity demand in 'MW'.
- We consider past 1 year trends only since peak demands and even off-peak hour demands have been increasing since the past 2 decades, and so keeping with current trends in data we would calculate incentives or penalties.
- Highest 10 percentile demand will be penalized more as compared to lesser demands.
- Any demand closer to median will not be penalized.
- Furthermore whenever demand is in lower percentiles - incentives in the form of less unit (Kwhr) price shall be offered to the consumer.

+ Code + Text

RAM Disk

Editing

[44] new_dataset.sort_values(by='Datetime')

26 to 50 of 8760 entries Filter ?

index	Datetime	MW	Decile	Outcome
112704	2017-08-11 01:00:00	1918.0	3	Incentive of: 10%
112705	2017-08-11 02:00:00	1784.0	2	Incentive of: 15%
112706	2017-08-11 03:00:00	1693.0	1	Incentive of: 20%
112707	2017-08-11 04:00:00	1648.0	1	Incentive of: 20%
112708	2017-08-11 05:00:00	1654.0	1	Incentive of: 20%
112709	2017-08-11 06:00:00	1723.0	2	Incentive of: 15%
112710	2017-08-11 07:00:00	1895.0	3	Incentive of: 10%
112711	2017-08-11 08:00:00	1997.0	4	Incentive of: 5%
112712	2017-08-11 09:00:00	2096.0	5	No penalty and no incentive
112713	2017-08-11 10:00:00	2205.0	7	Penalty of: 10%
112714	2017-08-11 11:00:00	2303.0	7	Penalty of: 10%
112715	2017-08-11 12:00:00	2400.0	8	Penalty of: 15%
112716	2017-08-11 13:00:00	2522.0	8	Penalty of: 15%
112717	2017-08-11 14:00:00	2580.0	9	Penalty of: 20%
112718	2017-08-11 15:00:00	2574.0	9	Penalty of: 20%
112719	2017-08-11 16:00:00	2548.0	9	Penalty of: 20%
112720	2017-08-11 17:00:00	2543.0	9	Penalty of: 20%
112721	2017-08-11 18:00:00	2564.0	9	Penalty of: 20%
112722	2017-08-11 19:00:00	2533.0	9	Penalty of: 20%

0s completed at 11:21 AM

As one can observe, after midnight and until early morning – there is less load demand on the grid and hence we are giving incentives to the user to charge their Electric vehicles at that time (as observed in first 8 points). As afternoon progresses and by the time of sunset – there is maximum demand on the grid and hence penalties in the range of 20 to 25% are imposed for EV charging.

FUTURE WORK

BLOCKCHAIN BASED

Ethereum Wallet

Currently the blockchain based Ethereum wallet is in development. Wallet based payment system would be integrated with the mobile application. This wallet system will allow users to manage their accounts on the Ethereum network. An Ethereum account is a type of account that can send transactions and keep track of its balance, with as many Ethereum addresses as it wants to send and receive funds, create smart contracts, interact with decentralized applications and more.

A public string of letters and digits that begins with "0x" is an Ethereum address. Every Ethereum address's balance is visible on the blockchain, but because each address on the network is represented by a string of numbers and characters, it is unknown who controls which address. Wallets are pieces of hardware or software that let users manage as many addresses as they need.

A private key, often known as a "password," will be used to control and secure Ethereum wallets and allow users to move their funds around. Only the wallet's creator should have access to these private keys because anybody with them can access their funds.

Data Exchange System for grid

Blockchain, in our view, is a distributed technology that increases confidence between several independent parties. It allows the creation of distributed peer-to-peer (P2P) networks, allowing community members to communicate with one another in a verified manner without the use of a reliable middleman. Microgrids can be strengthened by employing blockchain to manage transactions in a distributed database that takes the shape of a common accounting book.

This robustness may also be linked to the networks' protection against electronic attacks and their fortification on the solidarity front. These transactions involve the exchange of money, the exchange of electricity, or even the recording of power flows in the network.

MOBILE APPLICATION

It is required to enhance the user interface of current mobile application. Through which users can request for charging their EVs, and later for payment for the energy. The mobile application's backend is also currently being worked on. It also requires integration with Ethereum wallet and data exchange system built on the blockchain and the smart plug.

CONCLUSION

We have divided our project in 2 phases. As of now we are done with our 1st phase. In the first phase we are done with Load Curve Forecasting using Machine Learning, designed a smart plug and code interpretation for the smart plug circuit, also made an mobile application. In phase 2 we will work upon Blockchain based Ethereum wallet, information exchange using blockchain technology and enhance the mobile application. In the process of dealing with the project we came across to learn about various things like functioning of the equipment's which is used in the circuit, flutter, Blockchain, machine learning, design app UI, Jupyter Notebook , Solidity and skills acquired are team management, conflict management, cost management , research skills, communication, problem solving.

We are grateful and thankful to our mentor Prof. Dhiraj Bharat sir and whole Electrical department for the experiences and tutoring.

REFERENCES

For Arduino data sheet

<https://docs.arduino.cc/static/9d6ed041fec691039663ae42f50fabcc/A000066-datasheet.pdf>

for blockchain based platform

<https://www.sciencedirect.com/science/article/pii/S2405896320311241>

for ev smart charging

<https://driivz.com/blog/ev-smart-charging-benefits/>

LSTMs and Recurrent Neural Networks-

<https://wiki.pathmind.com/lstm>

<https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21>

Dataset Used-

<https://www.kaggle.com/robikscube/hourly-energy-consumption>

<https://www.mdpi.com/1424-8220/18/8/2575>

<https://create.arduino.cc/projecthub/nidhi17agarwal/uart-communication-between-arduino-uno-and-esp32-1170d5>