A
PBL REPORT
ON
"ARDUINO BASED SMART PHONE CHARGING CONTROLLER"

FOR PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE
PROJECT BASED LEARNING
OF S.E. E&TC – 2019 COURSE, SPPU, PUNE

BY:

1. **Kartik Madur (22145)** **Exam No.: S190053176**
2. **Jaidev Makode (22146)** **Exam No.: S190053179**
3. **Vedant Narawadkar (22151)** **Exam No.: S190053205**
4. **Nikhil Mahamuni (22152)** **Exam No.: S190053213**

Guide:
Mr. Sunil S. Khot

DEPARTMENT OF
ELECTRONICS & TELECOMMUNICATION ENGINEERING
PUNE INSTITUTE OF COMPUTER TECHNOLOGY
DHANKAWADI, PUNE – 43

A.Y. 2023-24

DEPARTMENT OF

ELECTRONICS & TELECOMMUNICATION ENGINEERING

PUNE INSTITUTE OF COMPUTER TECHNOLOGY

DHANKAWADI, PUNE – 43

- CERTIFICATE-

This is to certify that the Project Based Learning Report entitled

"ARDUINO BASED SMART PHONE CHARGING CONTROLLER"

Has been successfully completed by

1. Kartik Madur (22145)	**Exam No.: S190053176**
2. Jaidev Makode (22146)	**Exam No.: S190053179**
3. Vedant Narawadkar (22151)	**Exam No.: S190053205**
4. Nikhil Mahamuni (22152)	**Exam No.: S190053213**

Is a bonafide work carried out by them under the supervision of Mr. Sunil S. Khot and it is approved for the partial fulfillment of the requirements for the Project based learning of S.E. E&TC-2019 Course of the Savitribai Phule Pune University, Pune.

Mr. Sunil S. Khot	**Dr. M. V. Munot**
Project Guide	**HOD, E&TE**
Dept. Of E&TE	

Place: Pune

Date:

ACKNOWLEDGEMENT

We thank Mr. Sunil S. Khot sir for his continual guidance and support in the working of the project. We would also like to thank god for his continual eye on us and also answered all our prayers.

We express sincere gratitude to our Head of the department E&TE Dr. M. V. Munot for her leadership and constant motivation provided in successful completion of our academic semester. We record it as our privilege to deeply thank for providing us the efficient faculty and facilities to make our ideas into reality.

We express my sincere thanks to our project supervisor Mr. Sunil S. Khot for his novel association of ideas, encouragement, appreciation and intellectual zeal which motivated us to venture this project successfully.

Last but not the least we express our deep gratitude and affection to our parents who stood behind us in all our endeavors.

Place: Pune

Name of Students & Signs:

1. **Kartik Madur (22145)**
2. **Jaidev Makode (22146)**
3. **Vedant Narawadkar (22151)**
4. **Nikhil Mahamuni (22152)**

TABLE OF CONTENT

Chapter 1:

INTRODUCTION:

1.1 Background:

In recent years, the proliferation of portable electronic devices and renewable energy systems has increased the demand for efficient and reliable battery charging solutions. However, improper charging methods can lead to reduced battery life, decreased performance, and even safety hazards. To address these challenges, the development of smart charging controllers has become essential.

Arduino-based charging controllers offer a flexible and customizable platform for designing intelligent charging systems. These controllers leverage the versatility of Arduino microcontrollers and various input/output components to monitor and regulate the charging process effectively.

One crucial aspect of battery charging is controlling the charging duration to prevent overcharging. Overcharging can cause irreversible damage to the battery, leading to reduced capacity and potential safety risks. To mitigate this issue, incorporating a timer mechanism into the charging controller is essential.

The use of a rotary encoder provides a user-friendly interface for setting the charging timer. Rotary encoders offer precise and intuitive control, allowing users to adjust the timer duration with ease. Additionally, integrating an LCD display enhances user interaction by providing real-time feedback on the timer settings and charging status.

By combining these elements, the Arduino-based charging controller presented in this project aims to provide a reliable and user-friendly solution for charging various types of batteries. The ability to set a timer ensures that the charging process is controlled and optimized, ultimately extending battery life and improving overall performance.

1.2 Literature Survey:

Arduino-Based Charging Controllers: Several studies have explored the use of Arduino microcontrollers for developing charging controllers. M. Manikandan et al. (2018) demonstrated an Arduino-based solar charge controller that regulates the charging of batteries in solar power systems. Their system incorporates voltage and current sensors to monitor the charging process accurately. [1]

1. Timer-Based Charging Control: The integration of timer mechanisms in charging controllers has been investigated in various contexts. In their research, R. K. Singh et al. (2017) proposed a timer-based battery charging system for electric vehicles. The system utilizes a microcontroller to control the charging duration and ensure optimal charging without overcharging. [2]

2. Rotary Encoder Interface: The use of rotary encoders as user interfaces has been widely studied in the field of human-computer interaction. S. R. Mohanta et al. (2019) explored the application of rotary encoders in embedded systems for user input. Their work highlights the advantages of rotary encoders, such as precision control and intuitive operation. [3]

3. Battery Charging Optimization: Optimizing the charging process is crucial for maximizing battery life and efficiency. Y. Liu et al. (2016) proposed a dynamic charging algorithm for lithium-ion batteries to improve charging performance. Their algorithm adjusts the charging parameters based on battery state-of-charge and temperature. [4]

4. LCD Display Integration: LCD displays are commonly used in charging controllers to provide real-time feedback to users. A. Kumar et al. (2020) developed a smart battery charger with an LCD display for monitoring charging parameters. Their system enables users to track the charging progress and adjust settings as needed. [5]

5. Battery Management Systems: Battery management systems (BMS) play a crucial role in ensuring the safe and efficient operation of battery-powered devices. H. J. Bergveld et al. (2018) reviewed the principles and applications of BMS for various battery chemistries. Their

study emphasizes the importance of accurate monitoring and control in battery charging systems. [6]

6. Safety Considerations: Safety is a paramount concern in battery charging systems, especially when dealing with lithium-ion batteries. N. K. A. Ghani et al. (2021) investigated the safety aspects of lithium-ion battery charging and proposed strategies to mitigate risks, such as overcharging, thermal runaway, and short circuits. [7]

By reviewing the existing literature in these areas, valuable insights can be gained to inform the design and implementation of Arduino-based charging controllers with timer functionality, rotary encoder interfaces, LCD displays, and other key features. These insights can contribute to the development of more efficient, reliable, and user-friendly charging solutions for a wide range of applications.

1.3 Motivation:

This project is driven by several key motivations:

1. Battery Health and Longevity: Ensuring the longevity and health of batteries is crucial for various applications, from consumer electronics to renewable energy systems. Overcharging is a common issue that can significantly degrade battery life. By implementing a smart charging controller with a timer feature, the project aims to prevent overcharging and extend battery lifespan, reducing the need for frequent replacements and minimizing environmental impact.

2. User Empowerment and Convenience: Traditional charging systems often lack user-friendly interfaces and customization options. By integrating a rotary encoder and LCD display, the project empowers users to have precise control over the charging process. The ability to set a timer according to individual needs enhances convenience and ensures optimal charging without manual intervention.

3. Energy Efficiency and Sustainability: Optimal charging control not only preserves battery health but also improves overall energy efficiency. By automatically stopping the charging

process when the set timer expires, unnecessary energy consumption is avoided, leading to potential cost savings and reduced environmental footprint. This aligns with the growing emphasis on sustainability and energy conservation.

4. Educational and Innovative Value: Developing an Arduino-based charging controller provides a valuable opportunity for learning and innovation. Through hands-on experimentation and project development, individuals can gain practical insights into electronics, programming, and renewable energy concepts. The project encourages creativity and problem-solving skills, fostering a culture of innovation and exploration.

5. Versatility and Adaptability: The modular nature of Arduino-based systems allows for easy customization and adaptation to different charging requirements and battery types. Whether it's charging lead-acid batteries for automotive applications or lithium-ion batteries for portable electronics, the charging controller can be tailored to suit various scenarios.

In summary, the project is motivated by the desire to address the limitations of conventional charging methods, empower users with control and convenience, promote energy efficiency and sustainability, foster learning and innovation, and provide a versatile solution for intelligent battery charging.

1.4 Aim of the Project:

This project aims to develop an Arduino-based charging controller with timer functionality to address the limitations of conventional charging methods. By integrating a timer, rotary encoder, and LCD display, the project seeks to prevent overcharging, enhance user control, and promote energy efficiency in battery charging processes. Additionally, the project aims to provide a platform for learning, innovation, and customization, catering to diverse charging requirements and battery types. Ultimately, the goal is to deliver a practical and impactful solution for intelligent battery charging.

Chapter 2:

DESCRIPTION OF PROJECT:

2.1 Technical Approach:

The technical approach adopted for solving the problem of developing an Arduino-based charging controller with timer functionality involves several key steps and components:

1. Hardware Selection and Integration: The first step was to select suitable hardware components for the project. This included choosing an Arduino microcontroller board capable of interfacing with peripherals such as a rotary encoder and LCD display. Additionally, appropriate voltage regulation and charging circuitry were integrated to ensure compatibility with various battery types.

2. User Interface Design: The user interface plays a crucial role in facilitating user interaction and control. A rotary encoder was selected as the primary input device for setting the charging timer. Its rotary motion allows users to adjust the timer duration with precision, while the built-in push-button functionality enables additional input commands. An LCD display was incorporated to provide visual feedback on the timer settings and charging status, enhancing the user experience.

3. Software Development: The software for the Arduino-based charging controller was developed using the Arduino Integrated Development Environment (IDE) and the C/C++ programming language. The software includes routines for interfacing with the hardware components, implementing the charging control algorithm, and handling user input through the rotary encoder and LCD display. Special attention was given to optimizing the software for efficiency and reliability, ensuring smooth operation of the charging controller.

4. Testing and Validation: Finally, extensive testing and validation were conducted to verify the functionality and performance of the Arduino-based charging controller. This involved testing the controller with different battery types and charging scenarios to ensure compatibility

and reliability. Any issues or inconsistencies were identified and addressed through iterative refinement of the hardware and software components.

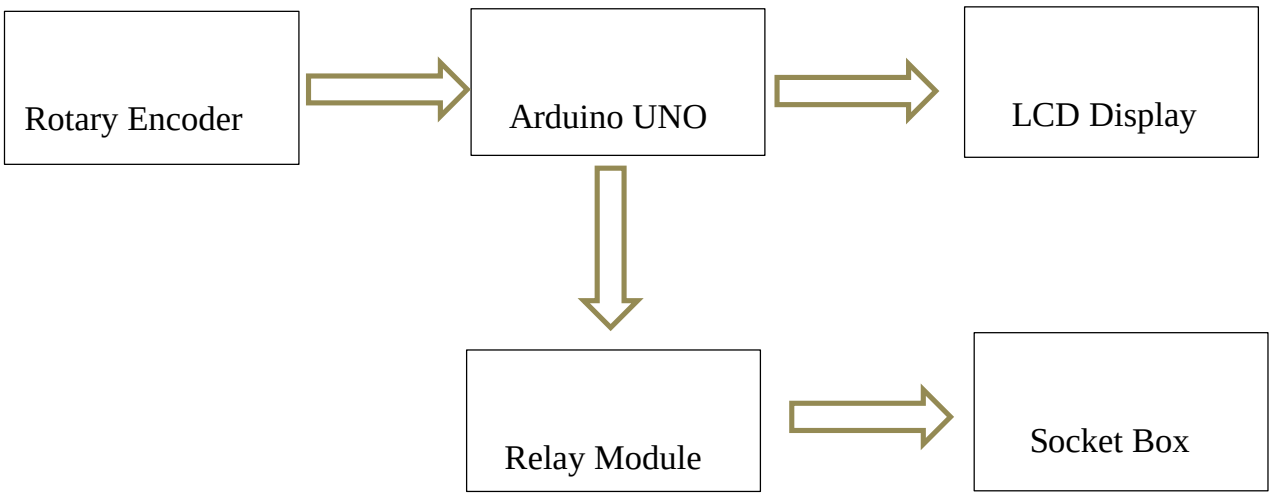
5. Charging Control Algorithm: A charging control algorithm was implemented to regulate the charging process based on the user-defined timer settings. The algorithm dynamically adjusts the charging duration to maintain optimal charging conditions while preventing overcharging. By continuously monitoring the battery voltage and current, the controller can adapt to changing battery conditions and ensure efficient and safe charging.

6. Safety Features Implementation: To ensure the safety of the charging process, additional safety features were implemented. These include overvoltage protection, overcurrent protection, and temperature monitoring. These safety mechanisms help prevent damage to the battery and ensure safe operation under various conditions.

By following this technical approach, the project aims to deliver a robust and user-friendly charging controller that effectively addresses the challenges of conventional charging methods while providing a platform for further customization and innovation.

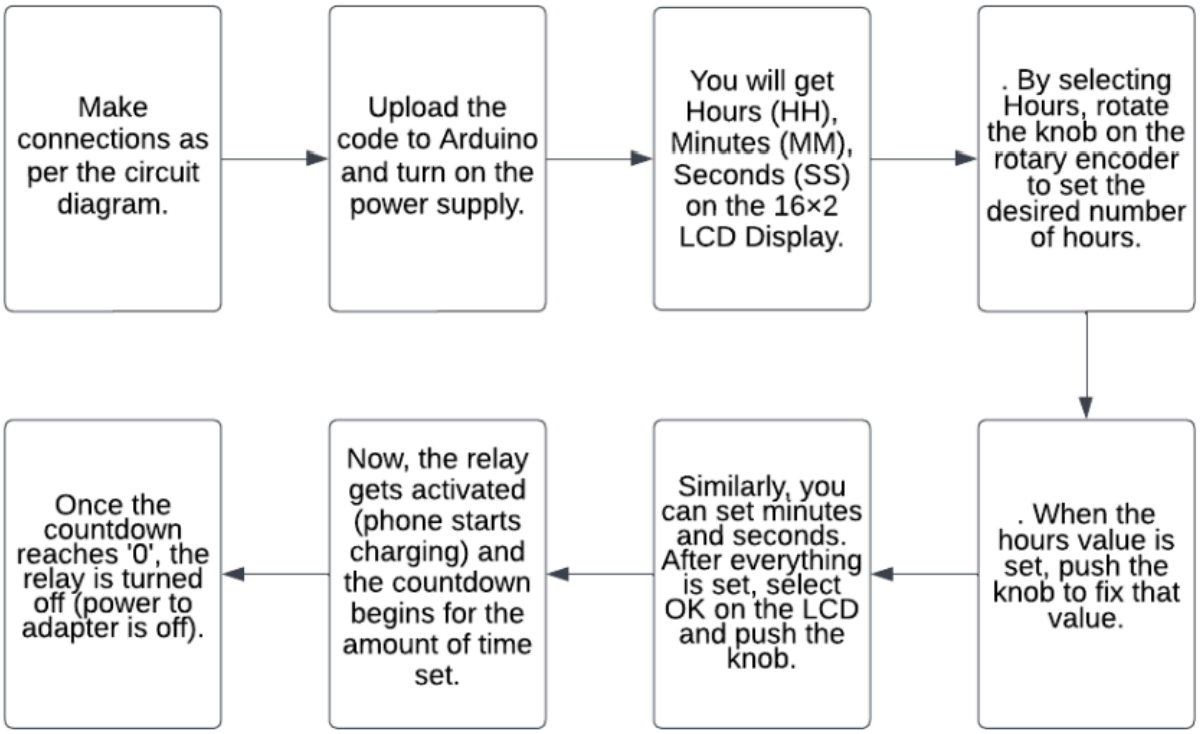
2.2 Block Diagram – 1:

This is the block diagram for the flow of input to output of the project:



2.3 Block Diagram – 2:

This is the block diagram for the flow of input to output of the project:



2.4 Hardware / Software Resources:

Hardware Resources:

1. Arduino Microcontroller Board: The project utilizes an Arduino microcontroller board as the central processing unit for the charging controller. The specific model chosen depends on the project requirements and compatibility with peripherals.

2. Rotary Encoder: A rotary encoder is employed as the primary input device for setting the charging timer. It provides precise control over timer adjustments through rotary motion and push-button functionality.

3. LCD Display: An LCD (Liquid Crystal Display) screen is integrated into the project to provide real-time feedback on the timer settings and charging status. The display enhances user interaction and visibility during the charging process.

4. Voltage Regulation Circuitry: To regulate the voltage supplied to the charging circuit and ensure compatibility with various battery types, voltage regulation circuitry is incorporated into the hardware design.

Software Resources:

1. Arduino Integrated Development Environment (IDE): The Arduino IDE serves as the primary software tool for developing and uploading code to the Arduino microcontroller board. It provides an intuitive interface for writing, compiling, and uploading code written in the C/C++ programming language.

2. C/C++ Programming Language: The project code is written in the C/C++ programming language, which is supported by the Arduino platform. This language is used to implement the charging control algorithm, interface with hardware components, and handle user input.

3. Arduino Libraries: Various Arduino libraries are utilized to simplify hardware interfacing and code development. These libraries provide pre-written functions and routines for interacting with peripherals such as rotary encoders, LCD displays, and sensors.

4. Serial Communication: Serial communication protocols such as UART (Universal Asynchronous Receiver-Transmitter) may be employed for debugging purposes or interfacing with external devices. Serial communication allows for data exchange between the Arduino board and a computer or other microcontrollers.

By leveraging these hardware and software resources, the project aims to implement a robust and efficient charging controller with timer functionality, providing users with a reliable solution for intelligent battery charging.

Chapter 3:

SYSTEM DESIGN:

3.1 Components:

1. Arduino UNO
2. 16×2 LCD Display
3. Rotary Encoder
4. 5V Relay Module
5. 10KΩ POT
6. Connecting Wires
7. Breadboard
8. Charging Adapter
9. Single Socket Power Outlet Box

Arduino UNO:

The Arduino Uno is an open-source micro controller based on the MicrochipATmega328P microcontroller and developed by Arduino. The board is equipped with sets of digital and analog input/output (I/O) pins that may be interfaced to various expansion boards (shields) and other circuits. The Arduino Uno is an open-source micro controller board based on the Microchip ATmega328P microcontroller and developed by Arduino.cc the board is equipped with sets of digital and analog input/output (I/O) pins that maybe interfaced to various expansion boards (shields) and other circuits

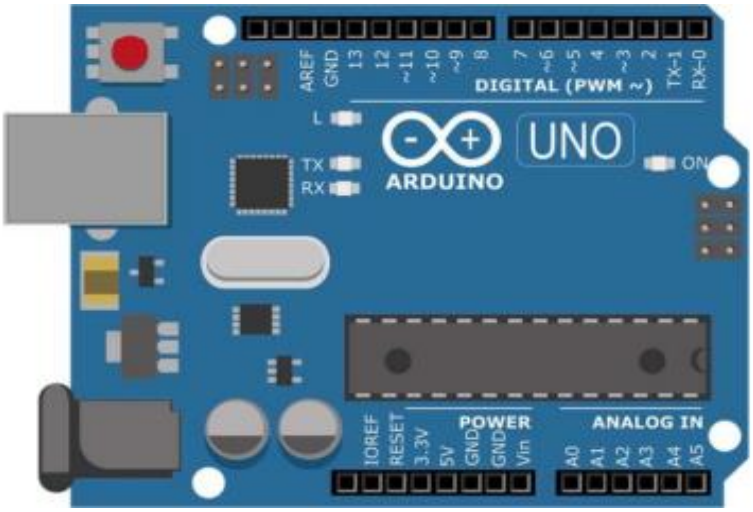


Fig. (1)

LCD Display:

Features of 16×2 LCD module:

1. Operating Voltage is 4.7V to 5.3V
2. Current consumption is 1mA without backlight
3. Alphanumeric LCD display module, meaning can display alphabets and numbers
4. Consists of two rows and each row can print 16 characters.
5. Each character is built by a 5×8 pixel box
6. Can work on both 8-bit and 4-bit mode
7. It can also display any custom generated characters

Fig. (2)

Rotary Encoder:

The rotary encoder is a crucial input device for the charging controller, allowing users to set the charging timer with precision. It comprises a rotating shaft and a push-button switch, generating electrical pulses as it rotates. This enables the Arduino microcontroller to detect rotation direction and speed.

Users can adjust the timer duration by rotating the encoder shaft, with each click representing a specific time increment or decrement. The push-button switch facilitates additional input commands, like confirming settings or initiating charging.

With its mechanical simplicity and durability, the rotary encoder ensures precise adjustments without wear. Interfaced with the Arduino, it captures user input for adjusting timer settings displayed on the LCD screen. Overall, the rotary encoder enhances user interaction, crucial for the charging controller's functionality and usability.

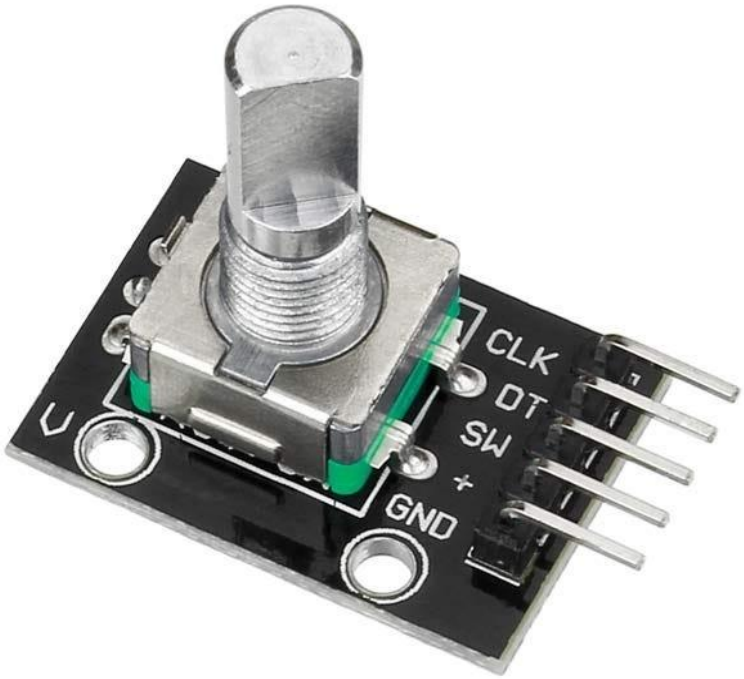


Fig. (3)

10KOHM POT:

Features

- Type: Rotary a.k.a Radio POT
- Available in different resistance values like 500Ω, 1K, 2K, 5K, 10K, 22K, 47K, 50K,
- 100K, 220K, 470K, 500K, 1 M.
- Power Rating: 0.3W
- Maximum Input Voltage: 200Vdc
- Rotational Life: 2000K cycles

Fig. (4)

Breadboard:

A breadboard is a rectangular plastic board with a bunch of tiny holes in it. These holes let you easily insert electronic components to prototype (meaning to build and test an early version of an electronic circuit, like this one with a battery, switch, resistor, and an LED (light emitting diode).

A breadboard is a platform you can use to build and test electronic circuits, usually without having do any soldering. Certain parts of the breadboard are wired together so that electricity can flow from component to component in orderly rows. Amateurs and experts alike use them to experiment with circuit ideas, and in some cases, they can be used to build useful devices directly on the breadboard.

A breadboard is usually covered with holes lined with metal, in which wires and electrical components such as resistors, diodes and capacitors can be plugged. The holes are divided into rows, and holes within particular rows are wired together on the underside of the breadboard so that an electric current can flow.

CHARGING ADAPTER:

An adapter is a device that converts attributes of one device or system to those of an otherwise incompatible device or system. Some modify power or signal attributes, while others merely adapt the physical form of one connector to another. An AC-to-DC power supply adapts electricity from household mains voltage (either 120 or 230volts AC) to low-voltage DC suitable for powering consumer electronics. Small, detached power supplies for consumer electronics are called, AC Adaptors or variously power bricks, wall warts, or chargers.

Single Socket power outlet box:

Power supplies are for safe isolation and power supply of circuits with power input up to 10 VA, for power supply of doorbells, house telephones, door openers, transducers, auxiliary circuits of contactors, lighting, relays, etc. A small metal or plastic junction box may form part of an electrical conduit or thermoplasticsheathed cable (TPS) wiring system in a building. If designed for surface mounting, it is used mostly in ceilings, under floors or concealed behind an access panel—particularly in domestic or commercial buildings. An appropriate type (such as that shown in the gallery) may be buried in the plaster of a wall (although full concealment is no longer allowed by modern codes and standards) or cast into concrete with only the cover visible.

3.2 Circuit Diagram:

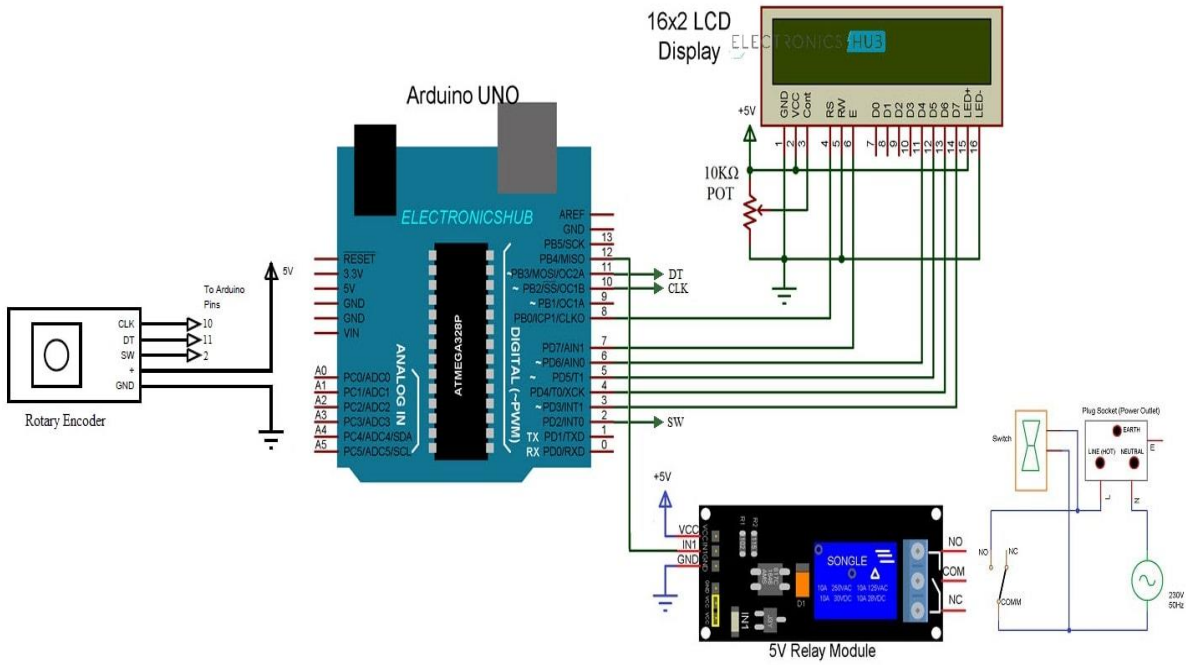


Fig. (5)

Chapter 4:

IMPLEMENTATION:

4.1 Implementation & Code:

Working:

After making the necessary connections as per the circuit diagram, upload the code to Arduino and turn on the power supply. You will get Hours (HH), Minutes (MM), Seconds (SS) on the 16×2 LCD Display. By selecting Hours, rotate the knob on the rotary encoder to set the desired number of hours. When the hours value is set, push the knob to fix that value.

Similarly, you can set minutes and seconds. After everything is set, select OK on the LCD and push the knob. Now, the relay gets activated (which means the phone starts charging) and the countdown begins for the amount of time set by you. Once the countdown reaches "0", the relay is turned off (power to adapter is off).

During charging if there is a power failure, the remaining time is stored in the memory and when the power comes back, it will prompt you whether to continue with the countdown or to set a new time. Accordingly, the charging will be performed.

Arduino Code:

```
#include<LiquidCrystal.h>
#include<EEPROM.h>
LiquidCrystal lcd (8,7,6,5,4,3);
#define outputA 10 //clk pin of rotary encoder
#define outputB 11 //dt pin of rotary encoder
#define button 2   //sw pin of rotary encoder
#define led 13     //status led
#define relay 12   //in1 of relay
```

```
int count = 1;
int current_state;
int previous_state;
int hh =0;
int mm=0;
int ss=0;
int h=0;
int m=0;
int s=0;
int ledToggle;
int previousState = HIGH;
unsigned int previousPress;
volatile int buttonFlag;
int buttonDebounce = 20;



void setup()
{
  lcd.begin(16,2);
  pinMode (outputA,INPUT);
  pinMode (outputB,INPUT);
  pinMode (button,INPUT_PULLUP);
  pinMode (led, OUTPUT);
  pinMode (relay, OUTPUT);
  digitalWrite(relay, HIGH);
  attachInterrupt(digitalPinToInterrupt(2), button_ISR,
CHANGE);
  Serial.begin (9600);
  previous_state = digitalRead(outputA);


    if(EEPROM.read(0)!=0 || EEPROM.read(1)!=0 ||
EEPROM.read(2)!=0)
```

```
{
  lcd.setCursor(0,0);
  lcd.print("Load Preset ?");
  lcd.print(" ");
  lcd.setCursor(0,1);
  lcd.print("      <NO>      ");
  int temp=5;
  while(temp>0 && ledToggle==0)
  {
    lcd.setCursor(14,0);
    lcd.print(temp);
    temp--;
    delay(1000);
  }
  if(temp==0 && ledToggle==0)
  {
   hh=EEPROM.read(0);
   mm=EEPROM.read(1);
   ss=EEPROM.read(2);
   timer();
  }
  else
  {
   EEPROM.write(0,0);
   EEPROM.write(1,0);
   EEPROM.write(2,0);
   ledToggle=0;
  }
}
lcd.clear();
lcd.setCursor(0,0);
lcd.print(" HH  MM  SS  OK ");
lcd.setCursor(0,1);
lcd.print(" 0   0   0      ");
```

```
}




void loop()
{
  encoder();
  if(count==1)
  {
    lcd.setCursor(0,0);
      lcd.print("<HH> MM  SS  OK ");
      while(count==1)
      {
        encoder();
        if(ledToggle)
        {
          h=1;
          m=0;
          s=0;
          lcd.setCursor(0,0);
          lcd.print(" HH  MM  SS  OK ");
          lcd.setCursor(1,1);
          lcd.print(hh);
          lcd.setCursor(5,1);
          lcd.print(mm);
          lcd.setCursor(9,1);
          lcd.print(ss);
          lcd.setCursor(0,1);
          lcd.print('<');
          lcd.setCursor(3,1);
          lcd.print('>');
```

```
          while(ledToggle)
          {
            encoder();
            lcd.setCursor(1,1);
            lcd.print(hh);
          }
        }
         EEPROM.write(0,hh);
         h=0;
         m=0;
         s=0;
        lcd.setCursor(0,0);
        lcd.print("<HH> MM  SS  OK ");
        lcd.setCursor(0,1);
        lcd.print(' ');
        lcd.setCursor(3,1);
        lcd.print(' ');
      }
}
else if(count==2)
 {
  lcd.setCursor(0,0);
    lcd.print(" HH <MM> SS  OK ");
    while(count==2)
    {
      encoder();
      if(ledToggle)
      {
        h=0;
        m=1;
        s=0;
        lcd.setCursor(0,0);
        lcd.print(" HH  MM  SS  OK ");
```

```
          lcd.setCursor(1,1);
          lcd.print(hh);
          lcd.setCursor(5,1);
          lcd.print(mm);
          lcd.setCursor(9,1);
          lcd.print(ss);
          lcd.setCursor(4,1);
          lcd.print('<');
          lcd.setCursor(7,1);
          lcd.print('>');

          while(ledToggle)
          {
            encoder();
            lcd.setCursor(5,1);
            lcd.print(mm);
          }
        }
        EEPROM.write(1,mm);
        h=0;
        m=0;
        s=0;
        lcd.setCursor(0,0);
        lcd.print(" HH <MM> SS  OK ");
        lcd.setCursor(4,1);
        lcd.print(' ');
        lcd.setCursor(7,1);
        lcd.print(' ');
      }
}
else if(count==3)
{
  lcd.setCursor(0,0);
    lcd.print(" HH  MM <SS> OK ");
```

```
while(count==3)
{
  encoder();
  if(ledToggle)
  {
    h=0;
    m=0;
    s=1;
    lcd.setCursor(0,0);
    lcd.print(" HH  MM  SS  OK ");
    lcd.setCursor(1,1);
    lcd.print(hh);
    lcd.setCursor(5,1);
    lcd.print(mm);
    lcd.setCursor(9,1);
    lcd.print(ss);
    lcd.setCursor(8,1);
    lcd.print('<');
    lcd.setCursor(11,1);
    lcd.print('>');
    while(ledToggle)
    {
      encoder();
      lcd.setCursor(9,1);
      lcd.print(ss);
    }
   EEPROM.write(2,ss);
   h=0;
   m=0;
   s=0;
  }
    lcd.setCursor(0,0);
    lcd.print(" HH  MM <SS> OK ");
    lcd.setCursor(8,1);
```

```
          lcd.print(' ');
          lcd.setCursor(11,1);
          lcd.print(' ');
      }
  }
  else if(count==4)
  {
    lcd.setCursor(0,0);
    lcd.print(" HH  MM  SS <OK>");
      while(count==4)
      {
        encoder();
        if(ledToggle)
        {
          timer();
        }
      }
  }
}


void encoder (void)
{
  current_state = digitalRead(outputA);
  if (current_state != previous_state)
  {
    if (digitalRead(outputB) != current_state)
    {
      if(count<4 && ledToggle==0)
      count ++;
      else
      {
        if(h==1)
        {
```

```
      if(hh<23)
      hh=hh+1;
    }
    else if(m==1)
    {
      if(mm<59)
      mm=mm+1;
    }
    else if(s==1)
    {
      if(ss<59)
      ss=ss+1;
    }
  }

}
else
{
  if(count>1 && ledToggle==0)
  count --;
  else
  {
    if(h==1)
    {
      if(hh>0)
      hh=hh-1;
    }
    else if(m==1)
    {
      if(mm>0)
      mm=mm-1;
    }
    else if(s==1)
    {
```

```
          if(ss>0)
          ss=ss-1;
        }
      }
    }
    Serial.print("Position: ");
    Serial.println(count);
  }
  previous_state = current_state;

}

void button_ISR()
{
   buttonFlag = 1;
   if((millis() - previousPress) > buttonDebounce &&
buttonFlag)
  {
    previousPress = millis();
    if(digitalRead(button) == LOW && previousState ==
HIGH)
    {
      ledToggle =! ledToggle;
      digitalWrite(led, ledToggle);
      previousState = LOW;
      //Serial.println(ledToggle);
    }

    else if(digitalRead(button) == HIGH && previousState
== LOW)
    {
      previousState = HIGH;
    }
    buttonFlag = 0;
```

```
  }
}

void timer (void)
{
  digitalWrite(relay, LOW);
  while(1)
  {
   lcd.setCursor(0,0);
   lcd.print("Charging...     ");
   lcd.setCursor(0,1);
   if(hh<10)
   {
    lcd.print('0');
    lcd.print(hh);
    lcd.print(':');
   }
   else
   {
    lcd.print(hh);
    lcd.print(':');
   }
   if(mm<10)
   {
    lcd.print('0');
    lcd.print(mm);
    lcd.print(':');
   }
   else
   {
    lcd.print(mm);
    lcd.print(':');
   }
   if(ss<10)
```

```
{
 lcd.print('0');
 lcd.print(ss);
 lcd.print("          ");
}
else
{
 lcd.print(ss);
 lcd.print("          ");
}
Serial.println(ss);
for(ss;ss>0;ss--)
{
   Serial.println(ss);
 if(ss<10)
 {
  lcd.setCursor(6,1);
  lcd.print('0');
  lcd.print(ss);
  EEPROM.write(2,ss);
 }
 else
 {
  lcd.setCursor(6,1);
  lcd.print(ss);
  EEPROM.write(2,ss);
 }
 delay(1000);
}
if(ss==0 && mm==0 && hh==0)
{
 lcd.clear();
 lcd.setCursor(0,0);
 lcd.print("Charged");
```

```
      digitalWrite(relay, HIGH);
      EEPROM.write(0,0);
      EEPROM.write(1,0);
      EEPROM.write(2,0);
      while(1);
     }
     ss=59;
     if(mm!=0)
     mm=mm-1;
     if(hh!=0)
     hh=hh-1;
     EEPROM.write(0,hh);
     EEPROM.write(1,mm);
     EEPROM.write(2,ss);

    }
}
```

Chapter 5:

RESULTS & CONCLUSION:

5.1 Results:

Once, the timer completes the input time set from the switch the mobile phone stops getting charged. This results in saving the battery life of the mobile phones

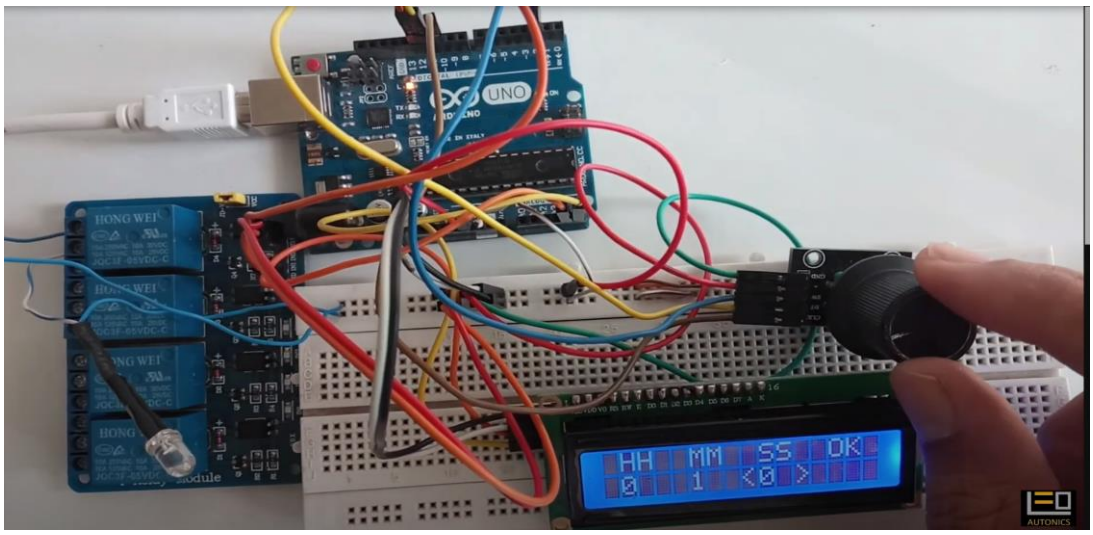


Fig. (6)

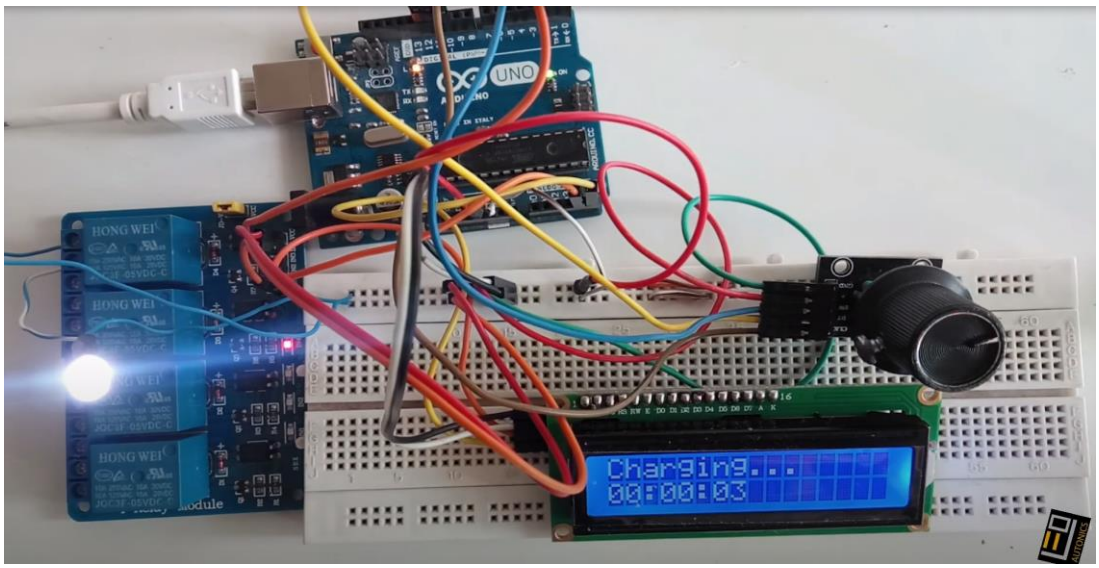


Fig. (7)

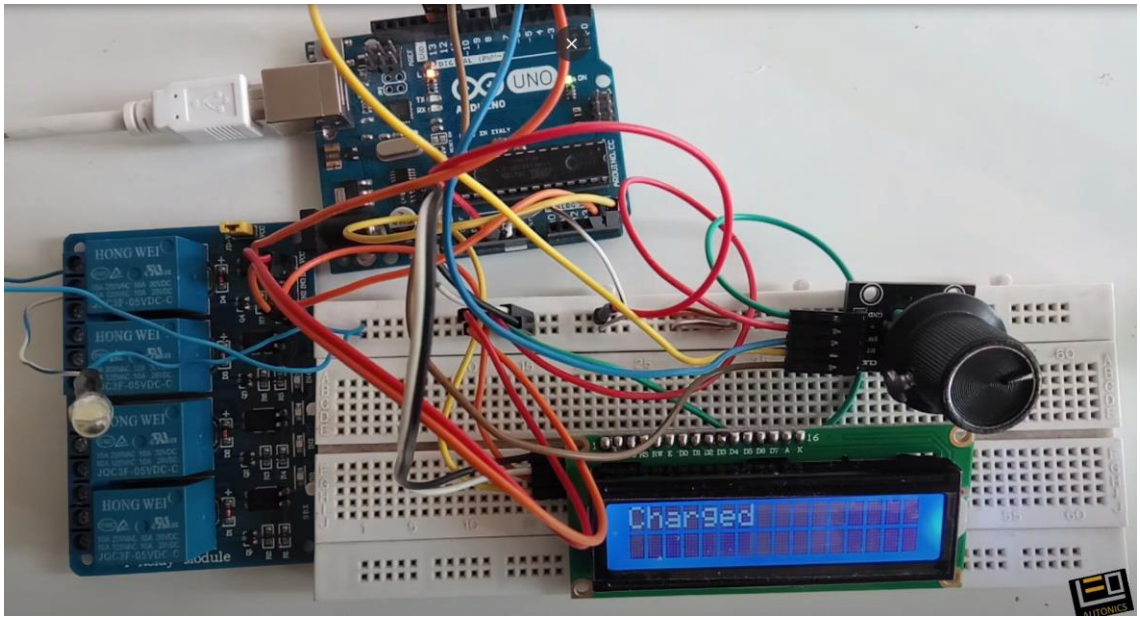


Fig. (8)

Advantages:

1. Smartphone Charging Controller project can be help full to control the charging time.
2. Improves battery life.
3. Very useful for people who tend to charge the phone during night time or those who often forget that they plugged in the phone to the charger.
4. Easy integration.
5. It is more reliable.
6. It requires less space.
7. Faster to connect and disconnect.
8. Multiple devices can be charged simultaneously

5.2 Conclusion & Future Scope:

Conclusion:

In conclusion, the Arduino-based charging controller developed in this project offers a versatile and user-friendly solution for intelligent battery charging. By integrating a rotary encoder, LCD display, and charging control algorithm, the project enables precise control over the charging process while preventing overcharging and ensuring battery longevity. This charging controller is designed to accommodate various battery types and charging scenarios, making it suitable for a wide range of applications, from portable electronics to renewable energy systems.

Overall, this project demonstrates the potential of Arduino-based systems in addressing real-world challenges and enhancing user experience in battery charging applications. Whether for personal use or integration into larger systems, the Arduino-based charging controller offers a practical and impactful solution for intelligent battery charging needs.

Future Scope:

The Arduino-based charging controller developed in this project lays the foundation for several potential avenues of future exploration and enhancement:

1. Advanced Charging Algorithms: Future iterations of the charging controller could incorporate advanced charging algorithms to optimize the charging process further. These algorithms may include temperature compensation, charge rate optimization, and adaptive charging profiles based on battery chemistry and condition.

2. Wireless Charging Integration: With the increasing popularity of wireless charging technology, integrating wireless charging capabilities into the controller could expand its compatibility and convenience. This would allow users to charge their devices wirelessly while still benefiting from the precision control and safety features of the Arduino-based controller.

3. Integration with Energy Storage Systems: Integrating the charging controller with energy storage systems, such as batteries or supercapacitors, could enable energy buffering and load management capabilities. This would allow for more efficient utilization of renewable energy sources and grid-tied systems, as well as backup power functionality during outages.

4. Commercialization and Mass Production: The charging controller could be further developed and refined for commercialization, targeting various market segments such as consumer electronics, automotive, and renewable energy industries. Mass production and distribution of the controller would make intelligent battery charging accessible to a wider audience, contributing to energy efficiency and sustainability efforts on a larger scale.

In summary, the future scope of the Arduino-based charging controller encompasses a range of possibilities for innovation and expansion. By exploring these avenues, the project can continue to evolve and adapt to emerging technologies and user needs, further enhancing its impact and relevance in the field of intelligent battery charging.

Chapter 6:

BILL OF MATERIALS:

Sr. No.:	Item:	Unit Price:	Units:	Cost:
1.	16×2 LCD Display	₹150	1	₹150
2.	Rotary Encoder	₹65	1	₹65
3.	5V Relay Module	₹120	2	₹240
4.	10k ohm POT	₹15	1	₹15
5.	Connecting Wires	₹50	1	₹50
6.	Single Socket Power Box	₹70	1	₹70
7.	PCB Board	₹40	1	₹40
	Total		**8**	**₹630**

Chapter 7:

REFERENCES:

[1] Rashid. M.H, "Power Electronics circuits devices and applications", Prentice Hall of India, New Delhi. 2004.

[2] Beechner, T., Sun, J.: Asymmetric Interleaving- a new approach to operate parallel converters. In Proc. of the IEEE Energy Conversion Congress and Exposition, San Jose, California, USA,pp.99-105(2009)

[3] Evans B.W., "Arduino Programming Notebook", http://playground.arduino.cc/uploads/Main/arduino_notebook_v1-1.pdf, August 2007.

[4] Charles Crawford. (2017). People too lazy to control their charging Retrieved from: https://www.lifehack.org/484043/7-reasons-smartphones-make-you-lazy

[5] Durfee W.," Arduino Microcontroller Guide', University of Minnesota, 2011.

[6] Brandon Jones (2017) Overheating: Is It a Charger Issue or a Phone Issue? Retrieved from: https://www.psafe.com/en/blog/overheating-charger-issue-phone-issue/

[7] ChatGPT by OpenAI: https://chat.openai.com/

[8] Arduino based Smartphone Charging Controller: https://www.electronicshub.org/arduino-based-smartphonecharging-controller/

[9] International Research Journal of Modernization in Engineering Technology and Science: file:///home/vlsilab/Downloads/fin_irjmets1655109373.pdf

[10] Arduino based Smartphone Charging Controller: (Reference Video) https://www.youtube.com/watch?v=rE6jkeakUEY