THE USE OF RFID CARD AS PAYMENT MEDIA IN LOCAL ELECTRICITY SYSTEM

A final project report
presented to
the Faculty of Engineering

By
Khoerrudin
002201400007

in partial fulfillment
of the requirements of the degree
Bachelor of Science in Electrical Engineering

President University
April 2018
DECLARATION OF ORIGINALITY

I declare that this final project report, entitled “The Use of RFID Card as Payment Media in Local Electricity System” is my own original piece of work and, to the best of my knowledge and belief, has not been submitted, either in whole or in part, to another university to obtain a degree. All sources that are quoted or referred to are truly declared.

Cikarang, Indonesia, April 2018

Khoerrudin
THE USE OF RFID CARD AS PAYMENT MEDIA IN LOCAL ELECTRICITY SYSTEM

By

Khoerrudin

002201400007

Approved by

Dr. –Ing Erwin Sitompul, S.T., M.Sc.
Final Project Supervisor

A. Suhartomo, Ph.D
Head of Study Program
Electrical Engineering
ACKNOWLEDGEMENT

In the name of Allah, the Entirely Merciful, the Especially Merciful, first and foremost I would like to express my gratitude to the almighty God Allah Subhanahu wa ta’ala., the merciful and the benevolent that has enabled me to complete and helped in all the time of research and writing of this final project report. Shalawat and Salam that I convey to Prophet Muhammad Solallahu’alaihi wasalam as a role models that has brought peace to the whole of nature. May conveyed also to his family, friends, and the whole ummah and hopefully we belong to it.

I also grateful to my family, my mother Mrs. Cucuh and my father Mr. Iman the most precious gift in my life who always being extremely supportive, praying, caring and help me to find strengths.

I am deeply thankful, to my final project supervisor Dr.–Ing. Erwin Sitompul, S.T., M.Sc. for his support, criticism and valuable assistant during my final project work. I am also very fortunate to carry out this research under his supervision because of his motivation, enthusiasm and knowledge.

Deep thanks are also due to all lecturers of Electrical Engineering for supporting and providing great assistance and the people who are inspirational and encouraging me along the way and who have made me what I am today.

Last but not least, all thanks and sincere gratitude are due to all electrical student batch 2014, the amazing people who have shared best memories with me. Best of luck for all of us now and in the future!.

Cikarang, April 2018

Khoerrudin
APPROVAL FOR SCIENTIFIC PUBLICATION

I hereby, for the purpose of development of science and technology, certify and approve to give President University a non-exclusive royalty-free right upon my final project report with the title:

THE USE OF RFID CARD AS PAYMENT MEDIA IN LOCAL ELECTRICITY SYSTEM

along with the related software or hardware prototype (if needed). With this non-exclusive royalty-free right, President University is entitled to conserve, to convert, to manage in a database, to maintain, and to publish my final project report. These are to be done with the obligation from President University to mention my name as the copyright owner of my final project report.

Cikarang, 10 April 2018

Khoerrudin
002201400007
Prepaid energy meter with RFID card as payment media is a unit that can be used to help people in charging electricity credit in their house. This unit will make people faster to top-up electricity credit that has been exhausted and can be controlled more easily to avoid the possibility of bills arrears and over consumption. This unit will not happen if the people fail to pay the bill by the noted invoice due date, the people will receive a disconnect notice and will be charged a late fee and will be disconnected if user do not pay the electric bill. With the top-up feature using RFID card, people do not need to buy out-of-home electricity credit because the energy meter will operate on charging by using RFID card. The RFID card stores a number of balance values which can be used as the input for the energy meter unit. The amount of balance will be converted to top-up balance. The amount of balance will be reduced incrementally in a proportion to the value of power usage. This system is succesfully designed and made the prototype uses microcontroller program (Arduino) as the main control of the unit.

*Keywords*: prepaid energy meter, RFID card, microcontroller.
TABLE OF CONTENT

DECLARATION OF ORIGINALITY .................................................................................. ii
APPROVAL PAGE ....................................................................................................... iii
ACKNOWLEDGEMENT ............................................................................................... iv
APPROVAL FOR SCIENTIFIC PUBLICATION ........................................................ v
ABSTRACT .................................................................................................................. vi
TABLE OF CONTENT ................................................................................................ vii
LIST OF FIGURES ...................................................................................................... ix
LIST OF TABLES ......................................................................................................... x

CHAPTER 1 INTRODUCTION .................................................................................. 1
  1.1. Background ........................................................................................................ 1
  1.2. Problem Statement ............................................................................................ 1
  1.3. Objectives .......................................................................................................... 2
  1.4. Scopes and Limitations ..................................................................................... 2
  1.5. Outlines ............................................................................................................ 2

CHAPTER 2 DESIGN REQUIREMENT AND SPECIFICATION ................................. 4
  2.1. Energy Meter ..................................................................................................... 4
  2.2. Microcontroller .................................................................................................. 5
    2.2.1. Arduino Uno ................................................................................................. 5
    2.2.2. Arduino Uno pins configuration ................................................................. 6
    2.2.3. Arduino IDE ............................................................................................... 8
  2.3. Modules ............................................................................................................. 9
    2.3.1. RFID Reader-MFRC522 ............................................................................. 9
    2.3.2. I2C LCD1602 ............................................................................................ 10
    2.3.3. One-Channel Relay Module ...................................................................... 11
  2.4. Interface ........................................................................................................... 12
    2.4.1. Matrix Membran Keypad 4x4 ................................................................. 12
    2.4.2. Liquid Crystal Display ............................................................................. 12
CHAPTER 3 DESIGN DEVELOPMENT AND IMPLEMENTATION .............................. 14
  3.1. Introduction to Design Implementations ........................................... 14
  3.2. Hardware Implementation ................................................................. 18
    3.2.1. Hardware Design ........................................................................ 18
    3.2.2. Hardware Components Construction ............................................. 19
  3.3. Software Implementation ..................................................................... 22
    3.3.1. Reading RFID Card .................................................................. 22
    3.3.2. Writing RFID Card ..................................................................... 23

CHAPTER 4 RESULTS AND ANALYSIS ............................................................ 24
  4.1. Results ................................................................................................. 24
    4.1.1. Construction of Top-up Unit ......................................................... 24
    4.1.2. Construction of RFID-based Prepaid Energy Meter Unit ................. 25
    4.1.3. Top-up the RFID Card ................................................................. 26
    4.1.4. Top-up The Prepaid Energy Meter ................................................ 28
    4.1.5. Operation of The Prepaid Energy Meter ........................................ 30
    4.1.6. LED Indicator of Energy Meter ..................................................... 32
  4.2. Analysis ................................................................................................ 33
  4.3. Strength and Weakness ........................................................................ 34

CHAPTER 5 CONCLUSIONS AND FUTURE DEVELOPMENTS ....................... 35
  5.1. Conclusions ......................................................................................... 35
  5.2. Future Developments .......................................................................... 35

REFERENCES .................................................................................................. 36

APPENDIX A CODING PROGRAM .................................................................. 37
  A.1. Prepaid Energy Meter Code ............................................................... 37
  A.2. Top-up Machine Code ....................................................................... 45
LIST OF FIGURES

Figure 2.1 Analog energy meter SMI-200S ................................................................. 4
Figure 2.2 Analog energy meter diagram ................................................................. 5
Figure 2.3 Arduino Uno ................................................................. 6
Figure 2.4 Arduino Uno pin configuration ............................................................... 8
Figure 2.5 Arduino IDE 1.8.5 interface ................................................................. 8
Figure 2.6 RFID principle system ................................................................. 9
Figure 2.7 RFID MFRC522 .................................................................................. 10
Figure 2.8 I2C LCD1602 module ........................................................................ 10
Figure 2.9 One-channel relay module .................................................................. 11
Figure 2.10 Keypad button 4x4 .......................................................................... 12
Figure 2.11 Liquid crystal display 16x2 ............................................................. 12
Figure 3.1 Block diagram of the top-up unit ....................................................... 14
Figure 3.2 Block diagram of the prepaid energy meter unit ..................................... 15
Figure 3.3 Flowchart of the top-up unit ................................................................ 16
Figure 3.4 Flowchart of the prepaid energy meter unit ........................................ 17
Figure 3.5 Hardware design of the RFID-based prepaid energy meter unit ......... 18
Figure 3.6 Hardware design of the top-up unit .................................................... 19
Figure 3.7 Schematic diagram of the prepaid energy meter unit ......................... 20
Figure 3.8 Schematic diagram of the top-up unit ................................................... 21
Figure 3.9 Schematic diagram for Prepaid Energy Meter based RFID .................. 21
Figure 4.1 Construction of the top-up unit .......................................................... 24
Figure 4.2 Construction of the RFID-based prepaid energy meter unit ............... 25
Figure 4.3 Reading the initial balance on the card ............................................... 26
Figure 4.4 The balance amount on the card is IDR 20 (14_{hex}) ......................... 27
Figure 4.5 The balance amount on the card is IDR 70 (46_{hex}) ......................... 28
Figure 4.6 Energy meter out of balance ............................................................. 29
Figure 4.7 Energy meter after top-up IDR 100 (64_{hex}) ..................................... 30
Figure 4.8 The decreasing of balance from 33.33 kWh to 17.33 kWh ................. 31
Figure 4.9 The last balance condition of the energy meter .................................... 32
LIST OF TABLES

Table 2.1 RFID Specifications .................................................................................................................. 10
Table 2.2 I2C LCD1602 Module Pin Configurations ............................................................................. 11
Table 2.3 LCD 1602 Pin Configurations ................................................................................................. 13
Table 3.1 List of Required Components .................................................................................................. 19
Table 3.2 Code of Reading Process ........................................................................................................ 22
Table 3.3 Code of Writing Process .......................................................................................................... 23
Table 4.1 Observation of Blinking on The Energy Meter ......................................................................... 32
Table 4.2 Theoretical Calculation of Blinking on The Energy Meter ......................................................... 33
Table 4.3 Comparison Data Between Theoretical Calculation and Observation ................................. 34
CHAPTER 1
INTRODUCTION

1.1. Background

Nowadays, RFID card is one of the greatest achievements in the world of information technology and becomes common in diverse society. RFID card is a type of card that can be used for payment instruments that progressively develop along with advances in microelectronics technology, especially in various types of payment services in Indonesia. The demands of society on this practical means of payment are increasing as their use is wider with the scope of its great application. Starting from the payment of transportation, shopping, restaurants and others.

Seeing this opportunity, the author hopes to implement a payment system using RFID cards that can be applied as a mean of payment of electrical energy. This can be beneficial for controlling the transaction between the producer and the user. This unit will not happen if the people fail to pay the bill by the noted invoice due date, the people will receive a disconnect notice and will be charged a late fee and will be disconnected if user do not pay the electric bill. By increasing the service in electricity consumption, the possibility of bills arrears and over consumption can be reduced. Because this unit can be controlled more easily by the user and there is no record of monthly usage by the officer. Of course, also it can reduce the cost of employees needed to manually check the consumption of electricity in every user.

1.2. Problem Statement

In order to avoid problem such as inability to top-up energy meter (for example, when the electricity is empty while the unique credit code from electricity store is not available 24 hours), at that point the author designed an RFID card top-up system so that the users are capable to top-up the electricity power credit by RFID card without having to use the unique credit code anymore. This gives the free access to utilize energy meter every time by simplifying the process of charging energy meter.
1.3. Objectives

The objectives of this final project are:
1. To design a prepaid energy meter which implements the RFID card in the purpose of top-up the energy meter, with Arduino Uno as the brain of the system.
2. Making the prototype of the prepaid energy meter and performing the testing and simulating to measure its performance.

1.4. Scopes and Limitations

This final project is conducted under the following scopes:
1. The final project is an improvement of the existing post-paid energy meter to be a pre-paid energy meter.
2. The final project uses an Arduino Uno microcontroller as the main control component.
3. The focus of the final project are the top-up energy meter using RFID card.

In conducting this research, there are limitations to be considered:
1. This device uses the card from the MRC522 RFID module.
2. The energy meter used is an analog energy meter SMI-200S, the measurement pulse is read and converted to digital data for further uses.
3. Money card to be used as top-up media. So, the stored value is the amount of money, instead the amount of kWh. This allows the card to be used as payment media of other transactions, not on for electricity.
4. The latest price of electricity (IDR/kWh) can be stored in the RFID card and the maximum time period when it should be updated, can be set.

1.5. Outlines

The final project report consists of five chapters and is outlined as follows:
1. Chapter 1 - Introduction
   This section will be a review of the general topic and the point of the undertaking. It comprises of: Final Project Background; Problem Statement; Final Project Objectives; Final Project Scopes and Limitations; and Final Project Outline.
2. Chapter 2 - Literature Review and Design Specifications
   This section will talk about the theoretical viewpoints prompting the execution of the final project and the details required by the specific equipment or programming in order to be able to have the capacity to satisfy the prescribed requirements and objectives yet calculations of required component values.

3. Chapter 3 - Design Implementations
   This section comprises of model and the detailed description of the utilized techniques and the implementation of the described design, for example, schematic outline, circuit outline, and programming codes.

4. Chapter 4 – Results and Analysis
   This section explains how the device functions. It additionally introduces information gathered from a few trials, results of the project and discussions. Simulation results are examined to finally conclude the strengths and weakness of the proposed system. This chapter will discuss further the result of Chapter III and discuss what actions taken from the analysis.

5. Chapter 5 - Conclusions and Recommendations
   This section comprises of conclusions obtained throughout the project implementation and suggestions for future improvement.
CHAPTER 2
DESIGN REQUIREMENT AND SPECIFICATION

2.1. Energy Meter

Energy meter is a device used to count the amount of power consumption used by residential, office, and industry. This energy meter is a type of energy meter based on a static meter analog system, the picture of analog energy meter is shown in Figure 2.1.

![Analog Energy Meter SMI-200S](image)

**Figure 2.1 Analog energy meter SMI-200S**

This type of analog energy meter will always be labelled with a number of Imp/kWh. Imp/kWh stands for Impressions per kWh (unit) of electricity passes through the energy meter where one ‘Impression’ is a short flash of calibration LED. The energy meter mentioned above is an energy meter labelled with 3200 Imp/kWh which means that the calibration LED will flash 3200 times for each kWh of electricity which passes through. The rate of the blinking of the calibration LED indicates how much power is currently passing through the energy meter [1].

In analog type energy meters, voltage and current values of each phase are obtained by voltage divider and current transformers respectively which are directly connected to the load as shown in Figure 2.2.
Figure 2.2 Analog energy meter diagram

Analog to digital converter converts these analog values to digitized samples and it is then converted to corresponding frequency signals by frequency converter. These frequency pulses then drive a counter mechanism where these samples are integrated over a time to produce the electricity consumption.

This final project modifies the analog energy meter and equips it with card reader device and RFID card as transaction device. The energy meter will operate based on the credit value entered into the energy meter, and then the credit value is used as a reference to control the operation of energy meter. The credit value will be reduced incrementally in proportion to the value of electrical energy that has been consumed (used).

2.2. Microcontroller

2.2.1. Arduino Uno

This final project uses an Arduino Uno microcontroller. Arduino Uno is a microcontroller that does not have too many pins and enough for creating this project. Arduino Uno microcontroller is an easy device to find in online or offline store with affordable price. The picture of Arduino Uno is shown in Figure 2.3.

Arduino Uno is a microcontroller board based the ATmega328P (datasheet). The microcontroller on the Arduino AVR based board has EEPROM: memory whose values are kept when the board is turned off (like a tiny hard drive). This library enables you to read and write those bytes. The ATmega328P on the Arduino Uno have amounts of EEPROM 1024 bytes.
Arduino Uno has 14 digital input or output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz quartz crystal, a USB connection cable or power with an AC-to-DC adapter or battery, a power jack, an ICSP header and a reset button [2], [3].

![Figure 2.3 Arduino Uno](image)

### 2.2.2. Arduino Uno pins configuration

Arduino Uno pin configuration has 3 parts which are power pins, analog pins, and digital pins with PWM (Pulse Width Modulation). Here are the descriptions of the Arduino Uno pins configuration:

a. Power Pins

Power pin gives power to another pin which require it. It can be activates the module or sensor used in the circuit by connecting 5V power pin to the Vcc pin of the module or sensor. In this descriptions Arduino Uno has several power pins as follows [2], [3]:

- 5V pin, the output voltage pin which gives 5 V.
- 3.3V pin, the output voltage pin which gives 3.3 V.
- GND (Ground) pin, there are 2 ground pins as the negative voltage pin.
- Vin pin, the input voltage to the board when it is running from external power.
- IOREF pin, it is a pin that provides the voltage reference with which the microcontroller operates.

b. Analog Pins

It is a specified analog pin with a 10-bit resolution that can be valued of 1024. The resulting integer values range from 0 to 1023 which can receive and sending analog value. In the voltage the value of analog pins is around 0 V until 5 V. There are 6 pins in it which is A0, A1, A2, A3, A4 and A5 [2], [3].
c. Digital Pins
There are 14 digital pins on the Arduino Uno that can be used as an input or output, using pinMode(), digitalWrite() and digitalRead() functions. They operate at 5 volts. Each pin can provide or receive 20 mA as recommended operating condition and has an internal pull-up resistor (disconnected by default) of 20-50 kilo ohm. A maximum of 40 mA is the value that must not be exceeded on any input or output pin to avoid permanent damage to the microcontroller. There are only 2 values in digital pin which is 1 and 0. 1 stands for HIGH it represents 5 volts output and 0 stands for LOW it represents 0 volts output. This 14 digital pins of Arduino Uno consists of pin number 0-13, and in the digital pins there are also contain PWM pins which it can be used as analog output and marked with symbol (~). The pins itself are 3, 5, 6, 9, 10, 11 [2], [3].

d. AREF pin
It is a pin that provides reference voltage for the analog inputs by used with analogReference(). By default, they measure from ground to 5 volts, though is it possible to change the upper end of their range [2], [3].

e. LED
There are 4 built-in LED in Arduino Uno board. There are ON LED, RX LED, TX LED and L LED. The ON LED is the indicator if there is voltage in to the Arduino. The RX LED is indicator if the Arduino receive signal from other Arduino parts, for example when the Arduino connect to the computer and the computer sending data to Arduino, the RX LED will turn ON because Arduino receiving data. TX LED is indicator if the Arduino transmit signal to the computer, for example when using Serial Monitor command in Arduino IDE, the TX LED will turn ON because Arduino is transmitting data. L LED is a built-in LED which connected to the pin 13 [2], [3].

f. Reset button
This button used to reset the program that working in the Arduino and makes it start again from the beginning [2], [3].

g. RX and TX pin
Receive (RX) pin is in the 0, the pin itself means receive TTL serial data. Transmit (TX) pin is in the 1, the pin itself means transfer TTL serial data. The picture of Arduino UNO pin configuration is shown in Figure 2.4 to make it easier understanding it [2], [3].
2.2.3. Arduino IDE

The software that the author used for this final project is Arduino IDE (Integrated Development Environment) 1.8.5. The interface of Arduino IDE can be seen in the Figure 2.5.

The Arduino IDE is a cross-platform Integrated Development Environment. This means can run it on every operating system. The Arduino IDE is a software based C-language program. This software is compatible to all Arduino microcontroller boards. In the software, tools menu contains the options of the board used. The users can create the code according to the project and upload it to the microcontroller board via USB cable [4].
2.3. Modules

2.3.1. RFID Reader-MFRC522

This module is a tool that is easy and good to use because it has a IC that can as a reader or writer is very integrated for contactless communication at 13.56 MHz. The RFID systems consist of three components in two combinations: a transceiver (transmitter/receiver) and antenna are usually combined as an RFID reader. A transponder (transmitter/responder) and antenna are combined to make an RFID tag. An RFID tag is read when the reader emits a radio signal that activates the transponder, which sends data back to the transceiver [5].

The RFID reader consist of a radio frequency module, a control unit and an antenna coil which generates high frequency electromagnetic field. On the other hand, the tag is usually a passive component, which consist of just an antenna and an electronic microchip, so when it gets near the electromagnetic field of the transceiver, due to induction, a voltage is generated in its antenna coil and this voltage serves as power for the microchip as shown in Figure 2.6.

The MFRC522 reader supports ISO / IEC 14443 A / MIFARE and NTAG. The internal MFRC522 transmitter is capable of pushing the reader / writer antenna designed to communicate with ISO / IEC 14443 A / MIFARE card / transponder and without additional active circuitry. In addition, the price for this tool is very affordable.

MFRC522 supports non-contact communication and uses higher MIFARE transfer speeds of up to 848 kBd in both directions [6]. But not all types of cards can be read by this tool because only certain types that can be read as above specifications. An illustration of the RFID module is shown in Figure 2.7.
Table 2.1 below summarized the full technical specifications of RFID Module:

Table 2.1 RFID Specifications

<table>
<thead>
<tr>
<th>Specification</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Working current</td>
<td>13 – 26 mA / DC 3.3 V</td>
</tr>
<tr>
<td>Standby current</td>
<td>10 – 13 mA / DC 3.3 V</td>
</tr>
<tr>
<td>Sleep current</td>
<td>&lt;80 uA</td>
</tr>
<tr>
<td>Peak current</td>
<td>&lt;30 mA</td>
</tr>
<tr>
<td>Working frequency</td>
<td>13.56 MHz</td>
</tr>
<tr>
<td>Card reading distance</td>
<td>0 – 60 mm</td>
</tr>
<tr>
<td>Protocol</td>
<td>SPI</td>
</tr>
<tr>
<td>Data communication speed</td>
<td>10 Mbit/s Max</td>
</tr>
<tr>
<td>Dimension</td>
<td>40 mm x 60 mm</td>
</tr>
<tr>
<td>Working temperature</td>
<td>-20 – 80 degree</td>
</tr>
<tr>
<td>Storage temperature</td>
<td>-40 – 85 degree</td>
</tr>
<tr>
<td>Humidity</td>
<td>5 % - 95 %</td>
</tr>
</tbody>
</table>

2.3.2. I2C LCD1602

The I2C LCD1602 module is a 2 line by 16 character display interfaced to an I2C daughter board. The author uses the I2C LCD1602 module to derive pin connections from microcontroller to the LCD. The Figure 2.8 is shown the I2C module of LCD.

Figure 2.7 RFID MFRC522

Figure 2.8 I2C LCD1602 module
This module is able to derive the LCD connections from 16 pin connections to become 4 pin connections. I2C LCD1602 module is the interface that only requires 2 data connections, +5 VDC and GND. This module is also equipped with a potentiometer that serves as a regulator of the light intensity of a displayed character. Table 2.2 shows the pin configurations [7].

### Table 2.2 I2C LCD1602 Module Pin Configurations

<table>
<thead>
<tr>
<th>Symbols</th>
<th>Types</th>
<th>Descriptions</th>
<th>Pins</th>
</tr>
</thead>
<tbody>
<tr>
<td>GND</td>
<td>Power</td>
<td>Supply &amp; Logic Ground</td>
<td>1</td>
</tr>
<tr>
<td>VCC</td>
<td>Power</td>
<td>Digital I/O 0 or Rx (serial receiver)</td>
<td>2</td>
</tr>
<tr>
<td>SDA</td>
<td>I/O</td>
<td>Serial Data Line</td>
<td>3</td>
</tr>
<tr>
<td>SCL</td>
<td>Clock</td>
<td>Serial Clock Line</td>
<td>4</td>
</tr>
</tbody>
</table>

### 2.3.3. One-Channel Relay Module

Relay module is a digital switch that has a connection pin as a control system controlled by the program to control higher voltages and currents. This module will be able to switch the current so it can flow or to cut it off. There is an induction part that reflects the input variables. When the induction is energized, the common terminal and normally open terminal will have continuity. The Figure 2.9 is shown the relay module.

![Figure 2.9 One-channel relay module](image)

This module can be used as a single chip microcontroller board module with single channel as an output contacts (Normally Close (NC), Normally Open (NO), Common port) and works as a Single Pole Double Throw (SPDT) switch. It also has Indicator light which are green for switch and red one for power. The module working in control voltage 5 VDC and it can reach 10 A maximum for DC load and 10 A maximum for AC load with maximum load voltage 240 V. It operates for High input voltage trigger and Low input voltage trigger as follow: VCC is connected to 5 V, GND is connected to ground and IN is connected to the digital pin which output is 3-5 V [8].
2.4. Interface

2.4.1. Matrix Membran Keypad 4x4

4x4 Keypad matrix is a combination of 16 buttons that have connections between the keys one with the other. There are 8 pins for connecting to microcontroller that consist of 4 pins for row and 4 pins for columns as can be seen in the Figure 2.10. The microcontroller will check each row and column when one of the keys is pressed either high or low at a moment. Underneath each key is a pushbutton, with one end connected to one row, and the other end connected to one column [9].

![Figure 2.10 Keypad button 4x4](image)

2.4.2. Liquid Crystal Display

LCD-1602 is one of the most commonly used display module especially for Arduino. It has 16 columns and 2 lines. It is equipped by blue or green backlight which contrast is adjustable. The screen resolution is 64.5 mm x 16 mm. LCD-1602 has 7 I/O pins which connect to the microcontroller. The Figure 2.9 is shown the LCD interface.

![Figure 2.11 Liquid crystal display 16x2](image)
The 16 pins are described below in Table 2.3 as a pin configurations of LCD [10]:

**Table 2.3 LCD 1602 Pin Configurations**

<table>
<thead>
<tr>
<th>Symbols</th>
<th>Descriptions</th>
<th>Pins</th>
</tr>
</thead>
<tbody>
<tr>
<td>VSS</td>
<td>Power supply (GND)</td>
<td>0</td>
</tr>
<tr>
<td>VDD</td>
<td>Power supply (+)</td>
<td>1</td>
</tr>
<tr>
<td>VO</td>
<td>Contrast adjustment</td>
<td>2</td>
</tr>
<tr>
<td>RS</td>
<td>Register select signal</td>
<td>3</td>
</tr>
<tr>
<td>RW</td>
<td>Data read/write</td>
<td>4</td>
</tr>
<tr>
<td>E</td>
<td>Enable signal</td>
<td>5</td>
</tr>
<tr>
<td>D0</td>
<td>Data bus line 0</td>
<td>6</td>
</tr>
<tr>
<td>D1</td>
<td>Data bus line 1</td>
<td>7</td>
</tr>
<tr>
<td>D2</td>
<td>Data bus line 2</td>
<td>8</td>
</tr>
<tr>
<td>D3</td>
<td>Data bus line 3</td>
<td>9</td>
</tr>
<tr>
<td>D4</td>
<td>Data bus line 4</td>
<td>10</td>
</tr>
<tr>
<td>D5</td>
<td>Data bus line 5</td>
<td>11</td>
</tr>
<tr>
<td>D6</td>
<td>Data bus line 6</td>
<td>12</td>
</tr>
<tr>
<td>D7</td>
<td>Data bus line 7</td>
<td>13</td>
</tr>
<tr>
<td>A</td>
<td>Power supply for LED B/L (+)</td>
<td>14</td>
</tr>
<tr>
<td>K</td>
<td>Power supply for LED B/L (-)</td>
<td>15</td>
</tr>
</tbody>
</table>
CHAPTER 3
DESIGN DEVELOPMENT AND IMPLEMENTATION

3.1. Introduction to Design Implementations

The purpose of this final project is to create a unit that can represent the efficient top-up transaction of electricity credit by using RFID card. This project is divided into two different units. The first is the unit to top-up the balance into the RFID card and the second is the energy meter unit that will be topped-up by the RFID card.

In this chapter, the implementation of the project will be described in block diagram followed by the flowchart of the whole system. Furthermore, the requirement of the components, the hardware design and the software implementation follow. The software implementation itself consists of programming of the main control for the circuit, created by Arduino IDE (Integrated Development Environment).

In the top-up system, the system focuses only on top-up RFID card. The RFID card will detect or read by the RFID reader/writer to know the amount of credit inside it. The Arduino will show the data on the LCD and will be processed to write data to the RFID card through the RFID reader/writer. Figure 3.1 shows the block diagram of the top-up unit.

![Figure 3.1 Block diagram of the top-up unit](image-url)
In the prepaid energy meter system, the rate of blinking from the energy meter will go to the Arduino as a power consumption signal. The Arduino will show the credit on the LCD and update every blink until the determine limit of credit to control the open and close of the relay. If the credit has reached the limit, the user has to be top-up by reading the data from the RFID reader to become the amount of credit that can control the relay. This happens if the card is detected by the RFID reader/writer and processed by the Arduino while waiting for the credit number option by the keypad. Figure 3.2 shows the block diagram of the prepaid energy meter.

![Figure 3.2 Block diagram of the prepaid energy meter unit](image)

Top-up the balance is done by inserting the RFID card into the RFID reader of the top-up unit, then entering the desired balance amount using keypad on the machine. When the RFID card has been read, the data to be shown on the LCD is the balance contained in the card. Afterwards, the new balance will be written on the RFID card. Block used in the RFID card to store the balance amount is block 34, which is in sector 8. Block 34 will always add up the previous balance amount with the new balance number entered.

The author also adds a top-up history on RFID cards so users will find it easier to know the top-up history of their cards. The top-up history of will be automatically written on blocks 33 and 32, in accordance with the amount of balances entered.
In addition to the top-up history of the RFID card, the author also adds the top-up history of on the energy meter. This top-up history will be automatically filled in blocks 30, 29, 28 which are on sector 7. In summary, the author's system is presented through the two parts of flowchart in Figure 3.3 and Figure 3.4.

Figure 3.3 Flowchart of the top-up unit
Figure 3.4 Flowchart of the prepaid energy meter unit
3.2. Hardware Implementation

3.2.1. Hardware Design

This energy meter unit is made of plastic box sized 18 cm x 11 cm x 6 cm with the analog energy meter above it. From Figure 3.5, it can be seen that this unit consists of various components and other parts such as LCD, 2 LED and keypad button. The microcontroller and the RFID reader are placed inside the box.

The top-up unit is approximately of the same size, with a number of different components. Figure 3.6 shows the details of the top-up unit.

Figure 3.6 shows the details of the top-up unit.
3.2.2. Hardware Components Construction

The list of required components is shown in the Table 3.1.

Table 3.1 List of Required Components

<table>
<thead>
<tr>
<th>Components</th>
<th>Descriptions</th>
<th>Amount</th>
</tr>
</thead>
<tbody>
<tr>
<td>Card Reader</td>
<td>RFID MFRC522 13.56MHz</td>
<td>2</td>
</tr>
<tr>
<td>Card</td>
<td>MIFARE 13.56MHz</td>
<td>2</td>
</tr>
<tr>
<td>Microcontroller</td>
<td>Arduino Uno ATmega328P</td>
<td>2</td>
</tr>
<tr>
<td>Display</td>
<td>Liquid crystal display 16x2</td>
<td>2</td>
</tr>
<tr>
<td>Module Display</td>
<td>I2C Liquid crystal display module</td>
<td>2</td>
</tr>
<tr>
<td>Resistor</td>
<td>220 Ohm 1/4Watt</td>
<td>4</td>
</tr>
<tr>
<td>Resistor</td>
<td>1 k Ohm 1/4Watt</td>
<td>2</td>
</tr>
<tr>
<td>Led</td>
<td>Light Emitting Diode 5Volt</td>
<td>5</td>
</tr>
<tr>
<td>Buzzer</td>
<td>Active Buzzer 5Volt</td>
<td>2</td>
</tr>
<tr>
<td>Electronic Controller</td>
<td>Matrix Membrane Keypad 4x4</td>
<td>2</td>
</tr>
<tr>
<td>Single Relay</td>
<td>SRD-05VDC-SL-C</td>
<td>1</td>
</tr>
<tr>
<td>Rectifier Diode</td>
<td>1n40007 1A 1000V</td>
<td>1</td>
</tr>
<tr>
<td>Energy Meter</td>
<td>Post-paid 900Rotary/kWh</td>
<td>1</td>
</tr>
<tr>
<td>Box</td>
<td>ABS Size 14x9x5cm and 18x11x6cm</td>
<td>1</td>
</tr>
<tr>
<td>Optocoupler</td>
<td>4N35</td>
<td>1</td>
</tr>
</tbody>
</table>
Figure 3.7 shows the complete schematic diagram of the proposed prepaid energy meter unit.

Figure 3.7 Schematic diagram of the prepaid energy meter unit
Figure 3.8 shows the complete schematic diagram of the top-up unit.
3.3. Software Implementation

This section explains the software implementation of Arduino IDE 1.8.5. Every program has 2 important parts. The first is the setup () function, which describes the functions that will be executed once the first time when the system is running. The second is the loop () function, which describes the function that will be executed in loop and run continuously. The complete lines of the program can be found on Appendix.

3.3.1. Reading RFID Card

The process of reading the card is one of the interesting parts of this project. In this section, the RFID card reading process on the device occurs when the card has been read by the card reader on a unit. This will happen if the position of the card is not too far from card reader and also not too quickly taken off before buzzer sounds once. Table 3.2 below shows code of reading process.

<table>
<thead>
<tr>
<th>Coding</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Serial.println(F(&quot; CURRENT DATA IN SECTOR:&quot;));</td>
<td>// Show the data on sector as it currently is sector 8 as a sector for Top-up and historical Top-up card</td>
</tr>
<tr>
<td>mfrc522.PICC_DumpMifareClassicSectorToSerial(</td>
<td></td>
</tr>
<tr>
<td>&amp;{(mfrc522.uid), &amp;key, Sector_1});</td>
<td></td>
</tr>
<tr>
<td>Serial.println();</td>
<td></td>
</tr>
<tr>
<td>status = (MFRC522::StatusCode)</td>
<td></td>
</tr>
<tr>
<td>mfrc522.MIFARE_Read(Current_Block, Buffer, &amp;size);</td>
<td></td>
</tr>
<tr>
<td>Serial.println(F(&quot; CURRENT BALANCE: &quot;));</td>
<td></td>
</tr>
<tr>
<td>dump_byte_array(Buffer, 16); Serial.println();</td>
<td></td>
</tr>
<tr>
<td>Serial.println();</td>
<td></td>
</tr>
<tr>
<td>mfrc522.PICC_DumpMifareClassicSectorToSerial(</td>
<td></td>
</tr>
<tr>
<td>&amp;{(mfrc522.uid), &amp;key, Sector_2});</td>
<td></td>
</tr>
<tr>
<td>Serial.println();</td>
<td></td>
</tr>
<tr>
<td>status = (MFRC522::StatusCode)</td>
<td></td>
</tr>
<tr>
<td>mfrc522.MIFARE_Read(Last_Block, Last, &amp;size);</td>
<td></td>
</tr>
<tr>
<td>Serial.println(F(&quot; LAST TOP-UP ENERGY METER: &quot;));</td>
<td></td>
</tr>
<tr>
<td>dump_byte_array2(Last, 16); Serial.println();</td>
<td></td>
</tr>
<tr>
<td>Serial.println();</td>
<td></td>
</tr>
<tr>
<td>status = (MFRC522::StatusCode)</td>
<td></td>
</tr>
<tr>
<td>mfrc522.MIFARE_Read(Last_Previous_Block, LastPrev, &amp;size);</td>
<td></td>
</tr>
<tr>
<td>Serial.println(F(&quot; LAST PREVIOUS TOP-UP ENERGY METER: &quot;));</td>
<td></td>
</tr>
<tr>
<td>dump_byte_array3(LastPrev, 16); Serial.println();</td>
<td></td>
</tr>
<tr>
<td>Serial.println();</td>
<td></td>
</tr>
</tbody>
</table>
3.3.2. Writing RFID Card

After the process of reading the card, the process of writing the RFID card is also an interesting part of this project. The process of writing RFID card is a process that makes the card filled with data. Then, this data can be used as parameter of balance amount in the RFID card. The writing of the data occurs when the RFID card is in the read position (the card is still attached to the card reader). This writing process can act as data addition or data reduction in accordance to the formula used. Table 3.3 below shows code of writing process.

<table>
<thead>
<tr>
<th>Coding</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>status = (MFRC522::StatusCode) mfc522.PCD_Authenticate(MFRC522::PICC_CMD_MF_AUTH_KEY_B, Header_Sector_1, &amp;key, &amp;(mfc522.uid));</td>
<td>// Authenticate using key B, the data on sector as it currently is sector 8 or not</td>
</tr>
<tr>
<td>Serial.println(F(&quot; NEW CURRENT BALANCE: &quot;)); dump_byte_array(Final, 16); Serial.println(); status = (MFRC522::StatusCode) mfc522.MIFARE_Write(Current_Block, Final, 16);</td>
<td>// Write data from sector 8 which is block number 34 as addition or deduction block</td>
</tr>
<tr>
<td>status = (MFRC522::StatusCode) mfc522.PCD_Authenticate(MFRC522::PICC_CMD_MF_AUTH_KEY_B, Header_Sector_2, &amp;key, &amp;(mfc522.uid));</td>
<td>// Authenticate using key B, the data on sector as it currently is sector 7 or not</td>
</tr>
<tr>
<td>status = (MFRC522::StatusCode) mfc522.MIFARE_Write(Last_Block, dataBlock, 16);</td>
<td>// Write data from sector 7 or 8 which is block number 34 or 30 historical last of usage or top-up block</td>
</tr>
<tr>
<td>status = (MFRC522::StatusCode) mfc522.MIFARE_Write(Last_Previous_Block, Last, 16);</td>
<td>// Write data from sector 7 or 8 which is block number 34 or 30 historical last of usage or top-up block</td>
</tr>
<tr>
<td>status = (MFRC522::StatusCode) mfc522.MIFARE_Write(Last_Last_Previous_Block, LastPrev, 16);</td>
<td>// Write data from sector 7 or 8 which is block number 34 or 30 historical last of usage or top-up block</td>
</tr>
<tr>
<td>if (status != MFRC522::STATUS_OK) { Serial.print(F(&quot;MIFARE_Write() failed: &quot;)); Serial.println(mfc522.GetStatusCodeName(status)); } Serial.println();</td>
<td>// The condition of the card is taken when the writing process runs. So, the process of writing will be failed</td>
</tr>
</tbody>
</table>
CHAPTER 4
RESULTS AND ANALYSIS

4.1. Results
This final project unit is designed to represent an efficient top-up transaction of electricity credit by using RFID card. This project is divided into two different units. The first is the unit for top-up the balance into RFID card and the second is the energy meter unit that will be top-up by the RFID card.

4.1.1. Construction of Top-up Unit
Almost all the components of the top-up unit are placed in the box in size 14 cm x 9 cm x 5 cm. The components such as LCD, LED and keypad are placed at the front side of the box to further show the aesthetics of component placement. For the card slot is placed at the topmost position.

Arduino microcontroller is mounted at the back of the box. It is so arranged to facilitate the user in maintaining the unit in case there is damage to the microcontroller. It is easy to troubleshoot if the other components have broken or trouble connections with the microcontroller.

The author puts the power jack at the bottom of the box. Figure 4.1 consists of pictures showing the construction of the top-up unit.

Figure 4.1 Construction of the top-up unit
4.1.2. Construction of RFID-based Prepaid Energy Meter Unit

The construction of the prepaid energy meter unit is almost the same as the construction of the top-up unit. All components are placed inside the box with some differences in position of the power jack and the card slot. For prepaid energy meter unit, the position of the power jack and the card slot are at the left and the right side of the box. The bottom side is used for the arrangement of the cable of the load.

In this prepaid energy meter unit, the author modified a post-paid energy meter into a prepaid energy meter by using microcontroller. The post-paid energy meter is placed at the top of the unit.

![Figure 4.2 Construction of the RFID-based prepaid energy meter unit](image)

Figure 4.2 Construction of the RFID-based prepaid energy meter unit
4.1.3. Top-up the RFID Card

Some simulations have been done to recharge the RFID card by simulating the initial zero condition. This experiment is carried out with different or varied top-up numbers. This is to show the changes in the balance amount of the RFID card and the history on the RFID card.

In this experiment, the author designed that the writing of the number of balances will be on the first 4 bytes of 16 total bytes contained in each block. Figure 4.3 shows the initial condition of balance amount of the RFID card before top-up.

![Figure 4.3 Reading the initial balance on the card](image)

Once the card is inserted into the top-up unit then it will read the balance in the card first. The balance is stored in block 34 (see red box). The value in this a block will continue to be added according to the balance amount on the RFID card.

In addition, the author adds a top-up history that is placed in block 33 (see blue box) as the last top-up of the RFID card and block 32 (see yellow box) as the top-up prior to the last top-up. So, the data written in block 32 is actually the data previously in block 33, while the data written in block 33 is the newest top-up data.

In the picture above, there is also data from sector 7, which were used to know the history of card usage in top-up the energy meter. The data will not change on the top-up card unit. The data will change when the card is inserted into the energy meter.
The concept of a charging history is: the newest data will be stored in the topmost block (block 30), and the last data will be stored at the bottom block (block 29). Figure 4.4 shows the balance after top-up the balance of IDR 20 ($14_{\text{hex}}$) into the RFID card.

![Figure 4.4 The balance amount on the card is IDR 20 ($14_{\text{hex}}$)](image)

Now the balance amount on the RFID card is IDR 20 ($14_{\text{hex}}$). The data on block 34 (red box) turns to $14_{\text{hex}}$ because the sum of the previous balance amounts with the amount of balances inserted which is ($0_{\text{hex}} + 14_{\text{hex}} = 14_{\text{hex}}$). The data on block 33 (blue box) turns to $14_{\text{hex}}$ because the amount entered is $14_{\text{hex}}$ (equal to IDR 20). While the data on block 32 (yellow box) changed to $32_{\text{hex}}$ because the previous data on block 33 is $32_{\text{hex}}$ (equal to IDR 50).

Finally, after adding the second new data to the card the data on block 34 (red box) turns to $46_{\text{hex}}$ because the amount of balances inserted is ($14_{\text{hex}} + 32_{\text{hex}} = 46_{\text{hex}}$). The data on block 33 (blue box) turns to $32_{\text{hex}}$ because the amount entered is $32_{\text{hex}}$ (equal to IDR 50). While the data on block 32 (yellow box) changed to $14_{\text{hex}}$ because the previous data on block 33 is $14_{\text{hex}}$ (equal to IDR 20).

This top-up process will keep repeating every time the user top-up in the card but can only see 2 previous transaction history which are block 33 (see blue box) and block 32 (see yellow box). Figure 4.5 shows the final condition of addition of balance amount into the card.
4.1.4. Top-up The Prepaid Energy Meter

Some simulations have been done to top-up the energy meter using RFID card. This experiment was conducted to see the condition of the decrease and addition of balances on energy meters with different or varying amounts of top-up. The purpose is to show changes in the number of RFID card balances.

The balance in the RFID card will decrease as the balance is filled to the energy meter and the balance data will be stored on the EEPROM. The EEPROM is also capable to record the last balance in case of power shutdown from PLN. The simple concept during top-up process is that the balance on the card will decrease while the EEPROM will increase.

In this unit, a decrease in the amount of balance occurs when the energy meter gets connection with the load by indicating through the blinking LED on the energy meter.

The minimum balance on the EEPROM of energy meter is IDR 2, if the balance is less than IDR 2 then the power line will be disconnected and the energy meter must be top-up immediately. Figure 4.6 shows the last balance condition of the energy meter.
The top-up process at the energy meter is the same as top-up process at the top-up unit. RFID card should be inserted into the provided slot and the energy meter will read the balance in the RFID card. The displayed balance originated from data in block 34 (see red box). The value in this block will continue to decrease in accordance with the amount of balances entered. The data will be transferred and stored to EEPROM and the data will be used as the balance on the energy meter.

After the successful top-up process, block 33 and 32 (the top-up history of the RFID card) will not change at all. The data that will change are the data in blocks 30, 29, and 28 (the usage history of the RFID card).

Here, the author adds a top-up history for the usage of balance amount on the energy meter. The data is placed in block 30 (see blue box) as the latest top-up while block 29 (see yellow box) as the previous top-up from the latest one and block 28 (see green box) as the oldest top-up prior.

The historical concept is the same as the previous top-up unit: the latest data will be stored in the topmost block, while the old data will go down to the bottom block. Figure 4.7 shows that the energy meter is already top-up with a balance of IDR 100 (64\text{hex}).
4.1.5. Operation of The Prepaid Energy Meter

In this section, the experiment is conducted to see the condition of the balance deduction in the energy meter that has been connected with the load. In the above experiments, the energy meter is already top-up with a balance of IDR 100 (64\text{\text{hex}}). When the load is connected to the energy meter, the LED on the energy meter will blink. The blinking LED indicates that the power has been used by the load. Every LED blinking, the balance on the energy meter will decrease.

Figure 4.8 shows the amount of balance in the energy meter decreasing from 33.33kWh (IDR 100) to 17.33kWh (IDR 52). This decreasing of balance happens quickly or slowly depends on how much power is currently passing through the energy meter. This affects the speed of blinking LED. If the power passing the energy meter getting bigger, then the LED will blink quickly, otherwise the LED will blink slowly.
The availability of balance on the energy meter will be displayed on the LCD. The author displays the balance amount in IDR and in kWh to make it easier for the user to see the amount of balance available. The availability of the balance on the energy meter is also indicated by the green LED that lights up under the LCD display.

If the amount of balance in the energy meter is less than IDR 15, then the display on the LCD will change and show that the balance amount is at low level and should be top-up immediately. The green LED will still be light up until the balance amount on the energy meter is at an amount less than equal to IDR 2.

If the amount of balance in the energy meter is less than equal to IDR 2, then the green LED will be off and the power will be cut off. the load connected to the energy meter will not work until the energy meter is re-top-up. Figure 4.9 shows the energy meter run out of balance.

Figure 4.8 The decreasing of balance from 33.33 kWh to 17.33 kWh
In chapter 2, it has been discussed that the energy meter has a calibration LED or pulse indicator for indicating whether the energy meter is connected to a load. Table 4.1 below shows the actual condition of the system when the loads is connected to the energy meter. Here, the author uses 2 different loads to provide different significant measuring and to ease the comparison and analysis.

**Table 4.1 Observation of Blinking on The Energy Meter**

<table>
<thead>
<tr>
<th>Load in Watts</th>
<th>Hairdryer</th>
<th>Laptop</th>
</tr>
</thead>
<tbody>
<tr>
<td>500 Watts</td>
<td>65 Watts</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Number of blinking (3 minutes measurements)</th>
<th>Hairdryer</th>
<th>Laptop</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. 23.33 blinks / minute</td>
<td>1. 3.33 blinks / minute</td>
<td></td>
</tr>
<tr>
<td>2. 23.33 blinks / minute</td>
<td>2. 3.67 blinks / minute</td>
<td></td>
</tr>
<tr>
<td>3. 23 blinks / minute</td>
<td>3. 3 blinks / minute</td>
<td></td>
</tr>
<tr>
<td>4. 23.67 blinks / minute</td>
<td>4. 3.33 blinks / minute</td>
<td></td>
</tr>
<tr>
<td>5. 23 blinks / minute</td>
<td>5. 3 blinks / minute</td>
<td></td>
</tr>
<tr>
<td>Average = 23.27 blinks/ minute</td>
<td>Average = 3.27 blinks / minute</td>
<td></td>
</tr>
</tbody>
</table>

**Figure 4.9 The last balance condition of the energy meter**

**4.1.6. LED Indicator of Energy Meter**

In chapter 2, it has been discussed that the energy meter has a calibration LED or pulse indicator for indicating whether the energy meter is connected to a load. Table 4.1 below shows the actual condition of the system when the loads is connected to the energy meter. Here, the author uses 2 different loads to provide different significant measuring and to ease the comparison and analysis.
Two experiments above have been done in measuring how the amount of blinking LED in certain time to compare with theoretical result. The above experiment indicates that the LED indication is not running constantly. There are interval differences in some conditions between one blink with the next one. Therefore, this data is important for the user to control their electricity consume.

4.2. Analysis

In this section, the results of experiment in previous section are compared with the theoretical calculation. The number of blinking LED is to be used as a measurement of power consumption. Table 4.2 below shows the data of blinking LED based on theoretical calculation.

<table>
<thead>
<tr>
<th>Load in Watts</th>
<th>Hairdryer</th>
<th>Laptop</th>
</tr>
</thead>
<tbody>
<tr>
<td>500 Watts</td>
<td>26.67 blinks / minute</td>
<td>3.47 blinks / minute</td>
</tr>
</tbody>
</table>

The above data is obtained by using the formula calculation as described below: here, the author uses 3200imp/kWh pulse rate of energy meter. This means that for 1 kWh outputs of power, the LED will blink for 3200 times. So, first we calculate the pulse rate for the loads by using the formula:

\[
Pulse\ rate = \left(\frac{Pulse\ Number\ for\ 1\ kWh\ x\ Load\ in\ kW}{60}\right)\ blink/\ minute \ldots \ldots \ldots (4.1)\]

The formula above calculates how many times the LED will blink in a certain time.

- Pulse rate for 500 watts in a minute:
  \[
Pulse\ rate = \left(\frac{3200 \times 0.5}{60}\right)\ blink/\ minute
  = 26.67\ blink/\ minute
  \]

- Pulse rate for 65 watts in a minute:
  \[
Pulse\ rate = \left(\frac{3200 \times 0.065}{60}\right)\ blink/\ minute
  = 3.47\ blink/\ minute
  \]
After this calculation is completed, the table below shows the difference between the practical data and the calculation data.

**Table 4.3 Comparison Data Between Theoretical Calculation and Observation**

<table>
<thead>
<tr>
<th></th>
<th>Hairdryer Practical data</th>
<th>Theoretical data</th>
<th>Laptop Practical data</th>
<th>Theoretical data</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Number of blinking</strong></td>
<td>1. 23.33 blinks / minute</td>
<td>26.67 blinks / minute</td>
<td>1. 3.33 blinks / minute</td>
<td>3.47 blinks / minute</td>
</tr>
<tr>
<td>(3 minutes measurements)</td>
<td>2. 23.33 blinks / minute</td>
<td></td>
<td>2. 3.67 blinks / minute</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3. 23 blinks / minute</td>
<td></td>
<td>3. 3 blinks / minute</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4. 23.67 blinks / minute</td>
<td></td>
<td>4. 3.33 blinks / minute</td>
<td></td>
</tr>
<tr>
<td></td>
<td>5. 23 blinks / minute</td>
<td></td>
<td>5. 3 blinks / minute</td>
<td></td>
</tr>
<tr>
<td><strong>Average</strong></td>
<td>= 23.27 blinks / minute</td>
<td></td>
<td>= 3.27 blinks / minute</td>
<td></td>
</tr>
<tr>
<td><strong>Power per hour</strong></td>
<td>436 watts</td>
<td>500 watts</td>
<td>61.31 watts</td>
<td>65 watts</td>
</tr>
</tbody>
</table>

If we look at the data from the above table, a very significant comparison only happens to the amount of blinking between theoretically and practically. But if calculated for electricity consumption there is only very slight difference. So, we can conclude that the difference between theory and practice is not significant.

### 4.3. Strength and Weakness

The strengths of the RFID-based prepaid energy meter proposed in this final project are:

- The simulated top-up using RFID card on the energy meter works well and smoothly.
- The measurement of power consumption by using LED indicator blinks works well.
  
The power consumed is proportional to the size of the connected load.

The weakness in this final project is:

- The maximum amount of top-up is only up to 85 kWh or equal to IDR 255 (0xFF hex).
CHAPTER 5
CONCLUSIONS AND FUTURE DEVELOPMENTS

5.1. Conclusions

Based on both experiments and analysis, there are several conclusions that could be author conclude:

1. The top-up unit and the prepaid energy meter unit is successfully designed. They use RFID as an input to top-up the amount of electricity balance, with Arduino Uno microprocessor as the brain of the system.
2. This overall unit has been successfully made. The prototype is tested and simulated. It functions as expected.

In addition, the author concludes one interesting point as follow: The amount of blinks provided by the energy meter has the difference between using theoretical and practical calculations. This happens because the load connected to the energy meter does not consume a constant power. For example, the hairdryer can only consume the maximum power 436 kWh which is approximately 12% less than the calculation data that should be 500 kWh.

5.2. Future Developments

Even though the objective of this final project is achieved successfully, the device can be improved to achieve better performance. Some recommendations for the prepaid energy meter with RFID card resulted from this final project are:

1. To improve the device performance, it is suggested to implement a voltage or current monitoring to show the historical power usage. Equipped with IoT feature. So, it can be easy for user to manage the electricity usage from outside their house.
2. The transaction billing system for balance credit can be connected with banking payment system. By this, it will be possible to use money card to top-up electricity.
REFERENCES


APPENDIX A
CODING PROGRAM

A.1. Prepaid Energy Meter Code

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>#include &lt;SPI.h&gt;</td>
<td>// SPI Library</td>
</tr>
<tr>
<td>#include &lt;EEPROM.h&gt;</td>
<td>// EEPROM Library</td>
</tr>
<tr>
<td>#include &lt;MFRC522.h&gt;</td>
<td>// RFID Library</td>
</tr>
<tr>
<td>#include &lt;Keypad.h&gt;</td>
<td>// Keypad Library</td>
</tr>
<tr>
<td>#include &lt;LiquidCrystal_I2C.h&gt;</td>
<td>// LCD Library with I2C</td>
</tr>
<tr>
<td>#define PULSEIN 8</td>
<td>// Pulse pin from energy meter</td>
</tr>
<tr>
<td>#define RELAY A3</td>
<td>// Relay Pin</td>
</tr>
<tr>
<td>#define RST_PIN 9</td>
<td>// Reset RFID Pin</td>
</tr>
<tr>
<td>#define SDA_PIN 10</td>
<td>// SDA RFID Pin</td>
</tr>
<tr>
<td>int Buzzer = A1;</td>
<td>// Buzzer Pin</td>
</tr>
<tr>
<td>int GreenLed = A2;</td>
<td>// GreenLed Pin</td>
</tr>
<tr>
<td>int a = 20, b = 50, c = 100;</td>
<td>// Set option number for storing data to EEPROM</td>
</tr>
<tr>
<td>String Button;</td>
<td>// Button declaration</td>
</tr>
<tr>
<td>float Units = 0;</td>
<td>// Set value of variable units</td>
</tr>
<tr>
<td>long Check = 0;</td>
<td>// Long type of number for checking the balance in card</td>
</tr>
<tr>
<td>unsigned int Rupiah = 0;</td>
<td>// Unsigned int type of number for balance conversion</td>
</tr>
<tr>
<td>byte Sector_1 = 8;</td>
<td>// Number of sector for topup in RFID Card</td>
</tr>
<tr>
<td>byte Sector_2 = 7;</td>
<td>// Number of sector for usage history in RFID Card</td>
</tr>
<tr>
<td>byte Current_Block = 34;</td>
<td>// Number of address for adding and deducting balance</td>
</tr>
<tr>
<td>byte Last_Block = 30;</td>
<td>// Number of address last history in RFID Card</td>
</tr>
<tr>
<td>byte Last_Previous_Block = 29;</td>
<td>// Number of address LastPrev history in RFID Card</td>
</tr>
<tr>
<td>byte Last_Last_Previous_Block = 28;</td>
<td>// Number of address LastLastPrev history in RFID Card</td>
</tr>
<tr>
<td>byte dataBlock1[] = { 0x14, 0x14, 0x14, 0x14, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00 };</td>
<td>// Number of data block for Top-up 20 amount</td>
</tr>
<tr>
<td>byte dataBlock2[] = { 0x32, 0x32, 0x32, 0x32, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00 };</td>
<td>// Number of data block for Top-up 50 amount</td>
</tr>
<tr>
<td>byte dataBlock3[] = { 0x64, 0x64, 0x64, 0x64, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00 };</td>
<td>// Number of data block for Top-up 100 amount</td>
</tr>
</tbody>
</table>
MFRC522::StatusCode status;
byte Header_Sector_1 = 35;
byte Header_Sector_2 = 31;
byte Buffer[18];
byte Current[18];
byte Final[16];
byte Last[18];
byte LastPrev[18];
byte LastLastPrev[18];
byte size = sizeof(Buffer);

const byte Header_Sector_1 = 35;
const byte Header_Sector_2 = 31;
const byte Buffer[18];
const byte Current[18];
const byte Final[16];
const byte Last[18];
const byte LastPrev[18];
const byte LastLastPrev[18];
const byte size = sizeof(Buffer);

const byte ROWS = 4;
const byte COLS = 3;
char keys[ROWS][COLS] =
{   {'1', '2', '3'},
    {'4', '5', '6'},
    {'7', '8', '9'},
    {'*', '0', '#'}
};

byte rowPins[ROWS] = {7, 6, A0, 5};
byte colPins[COLS] = {4, 3, 2};

Keypad keypad = Keypad(makeKeymap(keys), rowPins, colPins, ROWS, COLS);
MFRC522 mfrc522(SDA_PIN, RST_PIN);
 MFRC522::MIFARE_Key key;
LiquidCrystal_I2C lcd(0x3F, 16, 2);

void setup()
{
    Serial.begin(9600);
    while (!Serial);
    SPI.begin();
    mfrc522.PCD_Init();
    for (byte i = 0; i < 6; i++) {
        key.keyByte[i] = 0xFF;
    }
    pinMode(GreenLed, OUTPUT);
    pinMode(Buzzer, OUTPUT);
    pinMode(RELAY, OUTPUT);
    pinMode(PULSEIN, INPUT);
    digitalWrite(PULSEIN, HIGH);
    lcd.init();
    lcd.backlight();
    lcd.setCursor(1, 0);
    lcd.print("Prepaid Energy");
    lcd.setCursor(6, 1);
    lcd.print("Meter");
    delay(3000);
    lcd.clear();
    lcd.print("Intilizing...");
    delay(2000);
}
void loop()
{
  rfid_reader();
  rupiah = EEPROM.read(1);
  Units = Rupiah / 3.0;
  lcd.setCursor(0, 0);
  lcd.print("Units:");
  lcd.print(Units);
  lcd.setCursor(0, 1);
  if (Rupiah < 15)
  {
    lcd.print("LOW Balance:");
  } else
  {
    lcd.print("Balance:");
  }
  read_pulse();
  check_status();
}

void rfid_reader()
{
  char customKey = keypad.getKey();
  if (customKey) {
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("TopUp Amount:");
    lcd.setCursor(10, 1);
    lcd.print(Button);
    lcd.setCursor(6, 1);
    lcd.print("IDR");
    if (customKey == '#') {
      analogWrite(Buzzer, 255);
      analogWrite(GreenLed, 255);
      delay(1200);
      analogWrite(Buzzer, 0);
      analogWrite(GreenLed, 0);
    } else if (Button == "20") {
      TopUp(dataBlock1);
      Rupiah = EEPROM.read(1) + a;
      EEPROM.write(1, Rupiah);
      lcd.clear();
      lcd.setCursor(1, 0);
      lcd.print("TopUp Success!");
      delay(1500);
    } else if (Button == "50") {
      TopUp(dataBlock2);
      Rupiah = EEPROM.read(1) + b;
      EEPROM.write(1, Rupiah);
    } else if (Button == "100") {
      TopUp(dataBlock3);
      Rupiah = EEPROM.read(1) + c;
      EEPROM.write(1, Rupiah);
      lcd.clear();
      lcd.setCursor(1, 0);
      lcd.print("TopUp Success!");
      delay(1500);
    } else if (Button == "200") {
      TopUp(dataBlock4);
      Rupiah = EEPROM.read(1) + d;
      EEPROM.write(1, Rupiah);
      lcd.clear();
      lcd.setCursor(1, 0);
      lcd.print("TopUp Success!");
      delay(1500);
    } else if (Button == "500") {
      TopUp(dataBlock5);
      Rupiah = EEPROM.read(1) + e;
      EEPROM.write(1, Rupiah);
      lcd.clear();
      lcd.setCursor(1, 0);
      lcd.print("TopUp Success!");
      delay(1500);
    } else if (Button == "1000") {
      TopUp(dataBlock6);
      Rupiah = EEPROM.read(1) + f;
      EEPROM.write(1, Rupiah);
      lcd.clear();
      lcd.setCursor(1, 0);
      lcd.print("TopUp Success!");
      delay(1500);
    } else if (Button == "2000") {
      TopUp(dataBlock7);
      Rupiah = EEPROM.read(1) + g;
      EEPROM.write(1, Rupiah);
      lcd.clear();
      lcd.setCursor(1, 0);
      lcd.print("TopUp Success!");
      delay(1500);
    } else if (Button == "5000") {
      TopUp(dataBlock8);
      Rupiah = EEPROM.read(1) + h;
      EEPROM.write(1, Rupiah);
      lcd.clear();
      lcd.setCursor(1, 0);
      lcd.print("TopUp Success!");
      delay(1500);
    } else if (Button == "10000") {
      TopUp(dataBlock9);
      Rupiah = EEPROM.read(1) + i;
      EEPROM.write(1, Rupiah);
      lcd.clear();
      lcd.setCursor(1, 0);
      lcd.print("TopUp Success!");
      delay(1500);
    }
  }
  if (Button == "3") {
    analogWrite(Buzzer, 255);
    analogWrite(GreenLed, 255);
    delay(1200);
    analogWrite(Buzzer, 0);
    analogWrite(GreenLed, 0);
  }
  if (Button == "4") {
    analogWrite(Buzzer, 255);
    analogWrite(GreenLed, 255);
    delay(1200);
    analogWrite(Buzzer, 0);
    analogWrite(GreenLed, 0);
  }
  if (Button == "5") {
    analogWrite(Buzzer, 255);
    analogWrite(GreenLed, 255);
    delay(1200);
    analogWrite(Buzzer, 0);
    analogWrite(GreenLed, 0);
  }
  if (Button == "6") {
    analogWrite(Buzzer, 255);
    analogWrite(GreenLed, 255);
    delay(1200);
    analogWrite(Buzzer, 0);
    analogWrite(GreenLed, 0);
  }
  if (Button == "7") {
    analogWrite(Buzzer, 255);
    analogWrite(GreenLed, 255);
    delay(1200);
    analogWrite(Buzzer, 0);
    analogWrite(GreenLed, 0);
  }
  if (Button == "8") {
    analogWrite(Buzzer, 255);
    analogWrite(GreenLed, 255);
    delay(1200);
    analogWrite(Buzzer, 0);
    analogWrite(GreenLed, 0);
  }
  if (Button == "9") {
    analogWrite(Buzzer, 255);
    analogWrite(GreenLed, 255);
    delay(1200);
    analogWrite(Buzzer, 0);
    analogWrite(GreenLed, 0);
  }
  if (Button == "0") {
    analogWrite(Buzzer, 255);
    analogWrite(GreenLed, 255);
    delay(1200);
    analogWrite(Buzzer, 0);
    analogWrite(GreenLed, 0);
  }
}

void check_status()
{
  // The microcontroller will evaluate the input from pressed keypad according to the digit of the button. If the button is '20' then set the pin of Relay as HIGH and send 20 to EEPROM and write 20 deduction of amount of the data address in card. The relay is triggered. If the button is '50' then set the pin of Relay as HIGH and send 50 to EEPROM and write 50 deduction of amount of the data address in card. The relay is triggered. If the button is '100' then set the pin of Relay as HIGH and send 100 to EEPROM and write 100 deduction of amount of the data address in card. The relay is triggered and set up LCD clear with position of cursor on first row which will display text "TopUp Success!" and the position of cursor on second row which will display text "****" with delay 1.5 seconds after writing.
lcd.clear();
lcd.setCursor(1, 0);
lcd.print("TopUp Success!");
lcd.setCursor(6, 1);
lcd.print("****");
delay(1500);
} else if (Button == "100") {
    TopUp(dataBlock3);
    Rupiah = EEPROM.read(1) + c;
    EEPROM.write(1, Rupiah);
lcd.clear();
lcd.setCursor(1, 0);
lcd.print("TopUp Success!");
lcd.setCursor(6, 1);
lcd.print("****");
delay(1500);
} else {
    Button = "";
    lcd.clear();
lcd.setCursor(2, 0);
lcd.print("PLEASE TAKE");
lcd.setCursor(3, 1);
lcd.print("YOUR CARD");
delay(5000);
    loop();
} else if (customKey == '*') {
    mfrc522.PICC_HaltA();
    mfrc522.PCD_StopCrypto1();
} else {
    Button += customKey;
}
if (!mfrc522.PICC_IsNewCardPresent()) return;
if (!mfrc522.PICC_ReadCardSerial()) return;
lcd.clear();
lcd.setCursor(0, 0);
lcd.print("CURRENT BALANCE:");
lcd.setCursor(8, 1);
lcd.print("IDR");
lcd.setCursor(5, 1);
lcd.print(Current[0]);
delay(3000);
Serial.print(F("Card UID: "));
dump_byte_array(mfrc522.uid.uidByte, mfrc522.uid.size);
Serial.println();
Serial.print(F("PICC type: ")); MFRC522::PICC_Type piccType = mfrc522.PICC_GetType(mfrc522.uid.sak);
Serial.println(mfrc522.PICC_GetTypeName(piccType));
if (piccType != is successfully.

// Storage variable button
// Set up LCD clear
// Position of cursor on first row
// Display text "PLEASE TAKE"
// Position of cursor on second row
// Display text "YOUR CARD"
// Delay 5 seconds
// Recall void loop()
// Set up Information reset by pressing '*'
// Halt PICC
// Stop encryption on PCD

// Set up linear pressed keypad

// Look for new cards
// Terminate function above
// Select one of the cards
// Terminate function above

// Set up LCD clear
// Position of cursor on second row
// Display text "CURRENT BALANCE:" 
// Position of cursor on second row
// Display text "kWh"
// Position of cursor on second row
// Display text First address in datablock 27
// Delay 3 seconds

// Show some details of the PICC (that is: the tag/card, and type of card)

// Check for compatibility
// Authenticate data in card
status = (MFRC522::StatusCode) mfc522.PCD_Authenticate(MFRC522::PICC_CMD_MF_AUTH_KEY_A, Header_Sector_1, &key, &(mfrc522.uid)); Serial.println(F("Current data in sector:")); mfc522.PICC_DumpMifareClassicSectorToSerial(&(mfrc522.uid), &key, Sector_1); Serial.println(); // Display text in serial monitor “Current data in sector:”
status = (MFRC522::StatusCode) mfc522.MIFARE_Read(Current_Block, Buffer, &size); Serial.println(F("Current Balance:")); dump_byte_array(Buffer, 16); Serial.println(); Serial.println(); // Show the currently sector 8

// Reading data in serial monitor number of block Address which is 34
status = (MFRC522::StatusCode) mfc522.MIFARE_Read(Last_Block, LastPrev, &size); Serial.println(F("Last Top-up Data:")); dump_byte_array3(LastPrev, 16); Serial.println(); Serial.println(); // Display data in block 30 in serial monitor

// Reading data in serial monitor number of block Address which is 29
status = (MFRC522::StatusCode) mfc522.MIFARE_Read(Last_LastPrevious_Block, LastLastPrev, &size); Serial.println(F("Last LastPrevious Top-up Data:")); dump_byte_array4(LastLastPrev, 16); Serial.println(); Serial.println(); // Display data in block 29 in serial monitor

// Syntax function of dump_byte_array() for current condition
// State that Arduino will run the program inside the {} for size of byte < 18
void dump_byte_array(byte *Buffer, byte BufferSize) {
    for (byte i = 0; i < BufferSize; i++) {
        Serial.print(Buffer[i] < 0x10 ? " 0" : " ");
        Serial.print(Buffer[i], HEX);
    }
}
void dump_byte_array2(byte *Buffer, byte BufferSize) {
    for (byte i = 0; i < BufferSize; i++) {
        Serial.print(Buffer[i] < 0x10 ? " 0" : " ");
        Serial.print(Buffer[i], HEX);
        Last[i] = Buffer[i];
    }
}

void dump_byte_array3(byte *Buffer, byte BufferSize) {
    for (byte i = 0; i < BufferSize; i++) {
        Serial.print(Buffer[i] < 0x10 ? " 0" : " ");
        Serial.print(Buffer[i], HEX);
        LastPrev[i] = Buffer[i];
    }
}

void dump_byte_array4(byte *Buffer, byte BufferSize) {
    for (byte i = 0; i < BufferSize; i++) {
        Serial.print(Buffer[i] < 0x10 ? " 0" : " ");
        Serial.print(Buffer[i], HEX);
        LastLastPrev[i] = Buffer[i];
    }
}

void TopUp(byte dataBlock[]) {
    mfrc522.PICC_HaltA();
    mfrc522.PCD_StopCrypto1();
    add(dataBlock);
    if (Check < 0) {
        Serial.println("Your Balance is insufficient!!!");
        lcd.clear();
        lcd.setCursor(2, 0);
        lcd.print("Your Balance");
        lcd.setCursor(0, 1);
        lcd.print("Is Insufficient!");
        delay(2000);
    } else {
        if (! mfrc522.PICC_IsNewCardPresent())
            return;
        if (! mfrc522.PICC_ReadCardSerial())
            return;
        Serial.println("Top-Up Balance:");
        Serial.print(" IDR ");
        Serial.println(Button);
        Serial.println();
    }
}

// Equalization of array current and array buffer

// Syntax function of dump_byte_array() for last condition
// State that Arduino will run the program inside the {} for size of byte < 18
// Display in serial monitor the array in hexadecimal format
// Equalization of array Last and array buffer

// Syntax function of dump_byte_array() for last previous condition
// State that Arduino will run the program inside the {} for size of byte < 18
// Display in serial monitor the array in hexadecimal format
// Equalization of array LastPrev and array buffer

// Syntax function of dump_byte_array() for last last previous condition
// State that Arduino will run the program inside the {} for size of byte < 18
// Display in serial monitor the array in hexadecimal format
// Equalization of array LastLastPrev and array buffer

// Syntax function of topUp()
// Halt PICC
// Stop encryption on PCD
// Syntax for executing sum of write prev and new data

// State that if balance in card < than amount that will tapping to the machine
// Display in serial monitor “Your Balance is insufficient!”
// Set up LCD clear
// Position of cursor on first row
// Display text “Your Balance”
// Position of cursor on second row
// Display text “Is Insufficient!”
// delay in 5 seconds
// State that if balance in is >= than amount that will tapping to the machine
// Look for new cards
// Terminate function above

// Select one of the cards
// Terminate function above

// Display text "Top-Up Balance:" in serial monitor
// Display text "IDR:" in serial monitor
// Display button pressed in serial monitor
status = (MFRC522::StatusCode)
mfrc522.PCD_Authenticate(MFRC522::PICC_CMD_MF_AUTH_KEY_B, Header_Sector_1, &key, &(mfrc522.uid));
Serial.println(" New Current Balance:");
dump_byte_array(Final, 16);
Serial.println();
status = (MFRC522::StatusCode)
mfrc522.MIFARE_Write(Current_Block, Final, 16);
status = (MFRC522::StatusCode)
mfrc522.PCD_Authenticate(MFRC522::PICC_CMD_MF_AUTH_KEY_B, Header_Sector_2, &key, &(mfrc522.uid));
status = (MFRC522::StatusCode)
mfrc522.MIFARE_Write(Last_Block, Final, 16);
status = (MFRC522::StatusCode)
mfrc522.MIFARE_Write(Last_Previous_Block, Last, 16);
status = (MFRC522::StatusCode)
mfrc522.MIFARE_Write(Last_Last_Previous_Block, LastPrev, 16);
if (status != MFRC522::STATUS_OK) {
    Serial.print(F("MIFARE_Write() failed: "));
    Serial.println(mfrc522.GetStatusCodeName(status));
} Serial.println();
Serial.println(F("Current data in sector:"));
mfrc522.PICC_DumpMifareClassicSectorToSerial(& (mfrc522.uid), &key, Sector_1);
mfrc522.PICC_DumpMifareClassicSectorToSerial(& (mfrc522.uid), &key, Sector_2);
Serial.println();
mfrc522.PICC_HaltA();
mfrc522.PCD_StopCrypto1();
}

void add(byte dataBlock[]) {
for (int i = 0; i < 16; i++) {
    if (i < 4) {
        Final[i] = Current[i] - dataBlock[i];
    } else {
        Final[i] = dataBlock[i];
    }
}

long cur = Current[1];
long db = dataBlock[1];
Check = cur - db;

// Display new line in serial monitor

// Second Authenticated data in card before writing for sector 8

// Display in serial monitor "New Current Balance: 

// Execute syntax dump_byte_array to verify the number of array

// Writing state for block address number 34

// Second Authenticated data in card before writing for sector 7

// Writing state for block address number 30

// Writing state for block address number 29

// Writing state for block address number 28

// Check status read if not in OK Condition
// Display in serial monitor “MIFARE_Read() failed: ”

// Display in serial monitor "Current data in sector: ”

// Show the currently sector 8

// Show the currently sector 7

// Halt PICC
// Stop encryption on PCD

// Syntax function of add()
// State that Arduino will deduct amount of balance from current to the final with current data block minus datablock of desire

// Value of variable current array in second bytes
```c
void read_pulse()
{
    if (!digitalRead(PULSEIN))
    {
        if (Units < 1) {} else
            Units--; Rupiah = Units * 3.0;
        EEPROM.write(1, Rupiah);
        while (!digitalRead(PULSEIN));
    }
}

void check_status()
{
    if (Rupiah > 15)
    {
        analogWrite(RELAY, 255);
        analogWrite(GreenLed, 255);
    }
    if (rupiah <= 2)
    {
        analogWrite(RELAY, 0);
        analogWrite(GreenLed, 0);
    }
}
```

// Value of variable current array in second bytes
// Checking value of variable to indicate the amount of balance are fullfil to topup.

// Syntax of function read_pulse()
// State if energy meter did not send a signal to Arduino
// State if units < 1

// Set units decreased by formula
// Set formula for variable of rupiah
// EEPROM Write in address 1 of the amount rupiah
// State if energy meter did not send a signal to Arduino

// Syntax of function check_status()
// State if the rupiah > 15 the relay will close and the current will flow to the load and turning on the GreenLed to indicate the amount of balance still enough

// State if the rupiah <= 2 the relay will open and stop the current flow to the load and turning off the GreenLed to indicate the amount of balance has been exhausted
### A.2. Top-up Machine Code

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>#include &lt;SPI.h&gt;</td>
<td>// SPI Library</td>
</tr>
<tr>
<td>#include &lt;MFRC522.h&gt;</td>
<td>// RFID Library</td>
</tr>
<tr>
<td>#include &lt;LiquidCrystal_I2C.h&gt;</td>
<td>// LCD Library with I2C</td>
</tr>
<tr>
<td>#include &lt;Keypad.h&gt;</td>
<td>// Keypad Library</td>
</tr>
<tr>
<td>#define RST_PIN 9</td>
<td>// Define pin for RFID reset pin to pin no 9 in Arduino</td>
</tr>
<tr>
<td>#define SDA_PIN 10</td>
<td>// Define pin for RFID SDA pin to pin no 10 in Arduino</td>
</tr>
<tr>
<td>int RedLed = A1;</td>
<td>// RedLed Pin</td>
</tr>
<tr>
<td>int GreenLed = A2;</td>
<td>// GreenLed Pin</td>
</tr>
<tr>
<td>int Buzzer = A3;</td>
<td>// Buzzer Pin</td>
</tr>
<tr>
<td>long Check = 0;</td>
<td>// Set value of variable Check</td>
</tr>
<tr>
<td>String Button;</td>
<td>// Button declaration</td>
</tr>
<tr>
<td>byte Sector_1 = 8;</td>
<td>// Number of sector for topup in RFID Card</td>
</tr>
<tr>
<td>byte Sector_2 = 7;</td>
<td>// Number of sector for Usage history in RFID Card</td>
</tr>
<tr>
<td>byte Current_Block = 34;</td>
<td>// Number of address for adding and deducting balance</td>
</tr>
<tr>
<td>byte Last_Block = 33;</td>
<td>// Number of address Last history in RFID Card</td>
</tr>
<tr>
<td>byte Last_Previous_Block = 32;</td>
<td>// Number of address LastPrev history in RFID Card</td>
</tr>
<tr>
<td>byte dataBlock1[] = {</td>
<td>// Number of data block for Top-up 20 amount</td>
</tr>
<tr>
<td>0x14, 0x14, 0x14, 0x14, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00 }</td>
<td></td>
</tr>
<tr>
<td>byte dataBlock2[] = {</td>
<td>// Number of data block for Top-up 50 amount</td>
</tr>
<tr>
<td>0x32, 0x32, 0x32, 0x32, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00 }</td>
<td></td>
</tr>
<tr>
<td>byte dataBlock3[] = {</td>
<td>// Number of data block for Top-up 100 amount</td>
</tr>
<tr>
<td>0x64, 0x64, 0x64, 0x64, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00 }</td>
<td></td>
</tr>
<tr>
<td>MFRC522::StatusCode status;</td>
<td>// Tapping transfer data status</td>
</tr>
<tr>
<td>byte Header_Sector_1 = 35;</td>
<td>// Number of first row in sector 8</td>
</tr>
<tr>
<td>byte Header_Sector_2 = 31;</td>
<td>// Number of first row in sector 7</td>
</tr>
<tr>
<td>byte Buffer[18];</td>
<td>// Array Buffer in 18 size</td>
</tr>
<tr>
<td>byte Current[18];</td>
<td>// Array Current in 18 size</td>
</tr>
<tr>
<td>byte Final[16];</td>
<td>// Array Final on 16 size</td>
</tr>
<tr>
<td>byte Last[18];</td>
<td>// Array Last on 16 size</td>
</tr>
<tr>
<td>byte LastPrev[18];</td>
<td>// Array LastPrev on 16 size</td>
</tr>
<tr>
<td>byte size = sizeof(Buffer);</td>
<td>// Declaration of size for array buffer</td>
</tr>
<tr>
<td>const byte ROWS = 4;</td>
<td>// Number of rows on the keypad</td>
</tr>
<tr>
<td>const byte COLS = 3;</td>
<td>// Number of columns on the keypad</td>
</tr>
<tr>
<td>char keys[ROWS][COLS] = [</td>
<td>// Define the symbols on the buttons of the keypads</td>
</tr>
<tr>
<td>{'1', '2', '3'},</td>
<td></td>
</tr>
<tr>
<td>{'4', '5', '6'},</td>
<td></td>
</tr>
</tbody>
</table>
```cpp
byte rowPins[ROWS] = {8, 7, 6, 5};
byte colPins[COLS] = {4, 3, 2};

Keypad keypad =
Keypad(makeKeymap(keys),
rowPins, colPins, ROWS, COLS);

MFRC522 mfc522(SDA_PIN,
RST_PIN);
MFRC522::MIFARE_Key key;

LiquidCrystal_I2C lcd(0x3F, 16, 2);

void setup()
{
  pinMode(RedLed, OUTPUT);
  pinMode(GreenLed, OUTPUT);
  pinMode(Buzzer, OUTPUT);
  Serial.begin(9600);
  while (!Serial);
  SPI.begin();
  mfc522.PCD_Init();
  for (byte i = 0; i < 6; i++) {
    key.keyByte[i] = 0xFF;
  }
  lcd.init();
  lcd.backlight();
  lcd.setCursor(2, 0);
  lcd.print("PLEASE INSERT");
  lcd.setCursor(4, 1);
  lcd.print("YOUR CARD");
}

void loop()
{
  char customKey = keypad.getKey();

  if (customKey) {
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("TopUp Amount:");
    lcd.setCursor(6, 1);
    lcd.print("IDR");
    lcd.setCursor(10, 1);
    lcd.print(Button);
    if (customKey == '#') {
      analogWrite(A3, 255);
      analogWrite(A2, 255);
      delay(1200);
      analogWrite(A3, 0);
      analogWrite(A2, 0);
    }
  }
```

// Rows keypad pin connections

// Columns keypad pin connections

// Initialize an instance of class NewKeypad

// Define pin for RFID reset pin to pin no 9 in Arduino

// Define pin for RFID SDA pin to pin no 10 in Arduino

// Create MFRC522 instance.

// Mifare declaration card

// LCD interface pins

// Initialize serial communications with PC

// Do nothing if no serial port is opened (added for Arduinos based on ATmega32U4)

// Initialize SPI bus

// Initialize MFRC522 card

// Prepare the key (used both as key A and as key B) Using FFh which is the default at chip delivery from the factory

// Set up the LCD initialize

// Set up the LCD backlight

// Position of cursor on first row

// Display text “PLEASE INSERT”

// Position of cursor on Second row

// Display text “YOUR CARD”

// Main Loop

// Syntax to read pressed keypad

// State that if keypad pressed

// Set up LCD clear

// Position of cursor on first row

// Display text “TopUp Amount:”

// Position of cursor on second row

// Display text “IDR”

// Position of cursor on second row

// Display text "Pressed keypad"

// Syntax that if ‘#’ pressed

// Set pin A3 as HIGH/turn On(5v)

// Set pin A2 as HIGH/turn On(5v)

// Delay 1.2 seconds

// Set pin A3 as LOW/turn Off (5v)

// Set pin A2 as LOW/turn Off (5v)
if (Button == "20") {
  TopUp(dataBlock1);
  lcd.clear();
  lcd.setCursor(1, 0);
  lcd.print("TopUp Success!");
  lcd.setCursor(6, 1);
  lcd.print("****");
  delay(1500);
} else if (Button == "50") {
  TopUp(dataBlock2);
  lcd.clear();
  lcd.setCursor(1, 0);
  lcd.print("TopUp Success!");
  lcd.setCursor(6, 1);
  lcd.print("****");
  delay(1500);
} else if (Button == "100") {
  TopUp(dataBlock3);
  lcd.clear();
  lcd.setCursor(1, 0);
  lcd.print("TopUp Success!");
  lcd.setCursor(6, 1);
  lcd.print("****");
  delay(1500);
} else if (Button == "") {
  Button = "";
  lcd.clear();
  lcd.setCursor(2, 0);
  lcd.print("PLEASE TAKE");
  lcd.setCursor(3, 1);
  lcd.print("YOUR CARD");
  delay(5000);
  setup();
} else {
  Button += customKey;
}

if ( ! mfrc522.PICC_IsNewCardPresent()) return;
if ( ! mfrc522.PICC_ReadCardSerial()) return;
analogWrite(A3, 255);
analogWrite(A2, 255);
delay(800);
analogWrite(A3, 0);
analogWrite(A2, 0);
Serial.print(F("Card UID:"));
dump_byte_array(mfrc522.uid.uidByte, mfrc522.uid.size);
Serial.println();
Serial.print(F("PICC type: "));
MFRC522::PICC_Type piccType = mfrc522.PICC_GetType(mfrc522.uid.sak);
Serial.println(mfrc522.PICC_GetTypeName(piccType));

// The microcontroller will evaluate the input from pressed keypad according to the digit of the button. If the button is ‘20’ then write 20 addition of amount of the data address in card. If the button is ‘50’ then write 50 addition amount of the data address in card. If the button is ‘100’ then write 100 addition amount of the data address in card and set up LCD clear with position of cursor on first row which will display text “TopUp Success!” and the position of cursor on second row which will display text “****” with delay 1.5 seconds after writing is successfully.

// Storage pressed the button
// Set up LCD clear
// Position of cursor on first row
// Display text “PLEASE TAKE”
// Position of cursor on second row
// Display text “YOUR CARD”
// Delay 1.5 seconds
// Recall void setup()

// Set up linear pressed keypad

// Look for new cards
// Terminate function above
// Select one of the cards
// Terminate function above
// Set pin A3 as HIGH/turn On(5v)
// Set pin A2 as HIGH/turn On(5v)
// Delay 0.8 seconds
// Set pin A3 as LOW/turn Off (5v)
// Set pin A2 as LOW/turn Off (5v)

// Show some details of the PICC (that is: the tag/card, and type of card)

// Display in serial monitor “PICC type:”
if (piccType != MFRC522::PICC_TYPE_MIFARE_MINI && piccType != MFRC522::PICC_TYPE_MIFARE_1K && piccType != MFRC522::PICC_TYPE_MIFARE_4K) {
    Serial.println(F("MIFARE Classic cards only.");
    return;
}

status = (MFRC522::StatusCode) mfrc522.PCD_Authenticate(MFRC522::PICC_CMD_MF_AUTH_KEY_A, Header_Sector_1, &key, &(mfrc522.uid));
Serial.println(F("Current data in sector:"));
mfrc522.PICC_DumpMifareClassicSectorToSerial(&mfrc522.uid), &key, Sector_1);
Serial.println();

status = (MFRC522::StatusCode) mfrc522.MIFARE_Read(Current_Block, Buffer, &size);
Serial.print(F(" Current Balance:"));
dump_byte_array(Buffer, 16);
Serial.println(); Serial.println();

status = (MFRC522::StatusCode) mfrc522.MIFARE_Read(Last_Block, Buffer, &size);
Serial.print(F(" Last Top-up Data: Balance:"));
dump_byte_array2(Last, 16);
Serial.println(); Serial.println();

status = (MFRC522::StatusCode) mfrc522.MIFARE_Read(Last_Previous_Block, LastPrev, &size);
Serial.print(F(" Last Previous Top-up Data: Balance:"));
dump_byte_array3(LastPrev, 16);
Serial.println(); Serial.println();

mfrc522.PICC_DumpMifareClassicSectorToSerial(&mfrc522.uid), &key, Sector_2); Serial.println();
lcd.clear();
lcd.setCursor(0, 0);
lcd.print("CURRENT BALANCE:";)
lcd.setCursor(8, 1);
lcd.print("kWh");
lcd.setCursor(5, 1);
lcd.print(current[0]);
}
void dump_byte_array(byte *Buffer, byte BufferSize) {  
for (byte i = 0; i < BufferSize; i++) {  
Serial.print(Buffer[i] < 0x10 ? " 0" : " ");  
Serial.print(Buffer[i], HEX);  
Current[i] = Buffer[i];  
}  
}

void dump_byte_array2(byte *Buffer, byte BufferSize) {  
for (byte i = 0; i < BufferSize; i++) {  
Serial.print(Buffer[i] < 0x10 ? " 0" : " ");  
Serial.print(Buffer[i], HEX);  
Last[i] = Buffer[i];  
}  
}

void dump_byte_array3(byte *Buffer, byte BufferSize) {  
for (byte i = 0; i < BufferSize; i++) {  
Serial.print(Buffer[i] < 0x10 ? " 0" : " ");  
Serial.print(Buffer[i], HEX);  
LastPrev[i] = Buffer[i];  
}  
}

void TopUp(byte dataBlock[]) {  
mfrc522.PICC_HaltA();  
mfrc522.PCD_StopCrypto1();  
add(dataBlock);  
if (Check>255) {  
lcd.clear();  
lcd.setCursor(0, 0);  
lcd.print("Balance exceeds");  
lcd.setCursor(0, 1);  
lcd.print("maximum limit");  
delay(3000);  
} else {  
if ( ! mfrc522.PICC_IsNewCardPresent()) return;  
if ( ! mfrc522.PICC_ReadCardSerial()) return;  
Serial.println("Top-up Balance: ");  
Serial.println(" IDR ");  
Serial.println(Button);  
Serial.println();  
status = (MFRC522::StatusCode)  
mfrc522.PCD_Authenticate(MFRC522::  
PICC_CMD_MF_AUTH_KEY_B,  
Header_Sector_1, &key, &(mfrc522.uid));  
Serial.println(" New Current Balance");  
dump_byte_array(Final, 16);  
Serial.println();  
}

// Syntax function of dump_byte_array() for current condition
// State that Arduino will run the program inside the {} for size of byte < 18
// Display in serial monitor the array in hexadecimal format
// Equalization of array current and array buffer
// Syntax function of dump_byte_array2() for Last condition
// State that Arduino will run the program inside the {} for size of byte < 18
// Display in serial monitor the array in hexadecimal format
// Equalization of array Last and array buffer
// Syntax function of dump_byte_array3() for LastPrev condition
// State that Arduino will run the program inside the {} for size of byte < 18
// Display in serial monitor the array in hexadecimal format
// Equalization of array Last and array buffer
// Syntax function of topUp()
// Halt PICC
// Stop encryption on PCD
// Syntax for executing sum of write prev and new data
// State that if balance in card > than 255 that will not write again because already full
// Set up LCD clear
// Position of cursor on first row
// Display text "Balance exceeds"
// Position of cursor on second row
// Display text "maximum limit"
// delay in 3 seconds
// Look for new cards
// Terminate function above
// Select one of the cards
// Terminate function above
// Display in serial monitor "Top-up Balance:"
// Display in serial monitor " IDR"
// Display pressed Button
// Display new line
// Second Authenticated data in card before writing for sector 8
// Display in serial monitor "New Current Balance: ")
// Execute syntax dump_byte_array to verify the number of array
status = (MFRC522::StatusCode)
mfc522.MIFARE_Write(Current_Block, Final, 16);
status = (MFRC522::StatusCode)
mfc522.MIFARE_Write(Last_Block, dataBlock, 16);
status = (MFRC522::StatusCode)
mfc522.MIFARE_Write(Last_Previous_Block, Last, 16);

status = (MFRC522::StatusCode)
mfc522.PCD_Authenticate(MFRC522::PICC_CMD_MF_AUTH_KEY_B, Header_Sector_2, &key, &(mfc522.uid));

if (status != MFRC522::STATUS_OK) {
  Serial.print(F("MIFARE_Write() failed: "));
  Serial.println(mfc522.GetStatusCodeName(status)); } Serial.println();

Serial.println(F("Current data in sector:");
mfc522.PICC_DumpMifareClassicSectorToSerial(& (mfc522.uid), &key, Sector_1);
mfc522.PICC_DumpMifareClassicSectorToSerial(& (mfc522.uid), &key, Sector_2);
Serial.println();

mfc522.PICC_HaltA();
mfc522.PCD_StopCrypto1();
}

void add(byte dataBlock[]) {
  for (int i = 0; i < 16; i++) {
    if (i < 4) {
      Final[i] = Current[i] + dataBlock[i];
    } else {
      Final[i] = dataBlock[i];
    }
  }
  long cur = Current[1];
  long db = dataBlock[1];
  Check = cur + db;
}