Department of Electrical and Electronic Engineering
Faculty of Engineering
Daffodil International University
Dhaka 1341, Bangladesh

Project Report

Overload Protection For A Classroom

Course Code: EEE 458
Course Title: Power system Protection Laboratory
Instructor: Kanij Ahmad
Designation: Senior Lecturer, Department of EEE

Grade:

Submission Date: 05/12/2022

Submitted By

1. Md. Hasemi Rafsan Jani Shohan 191-33-849
2. Milon Hossain 191-33-847
3. Md. Mehedi Hasan 191-33-837
4. Utshab Biswas 191-33-863
5. Md. Arifuzzaman 191-33-883
6. Dewan Md. Nahid 191-33-882
7. Aliraz 191-33-885
8. Eya Muhasi Vuban 191-33-876

Remarks

INTRODUCTION:

Overload protection is a protection against a running overcurrent that would cause overheating of the protected equipment. Hence, an overload is also type of overcurrent. Overload protection module prevent circuit damage by monitoring the current in the circuit and breaking the circuit when an electrical overload or a phase failure is detected. Since overload protection module are much cheaper than electrical equipment, they provide an affordable way of protection.

When overload happened in a circuit, if the circuit is not connected by a protective device such as a circuit breaker, relay etc., a large amount of current will flow through the circuit due to overload. We know that each wire has a certain current carrying capacity. Excessive current flow can melt the wire insulation and may cause fire. Overload can cause financial losses by damaging our valuable electrical equipment.

There are different types of overload protection. Examples,
1. Relay based overload protection
2. Microcontroller or electronic based overload protection

In our project for overload protection we used microcontroller based or electronic overload protection module.

EQUIPMENTS:

1. Arduino UNO
2. Microcontroller
3. I²CDriver
4. LCD Display
5. Current Sensor
6. Relay
7. LED Bulb
8. Push Button Switch
9. Connecting wires
10. Breadboard

Picture1: Arduino UNO

Picture2: Microcontroller

Picture3: LCD Display & Driver

Picture4: Current Sensor

Picture5: Relay

Picture6: LED Bulb

Picture7: Push Button Switch

Picture5: Connecting wires

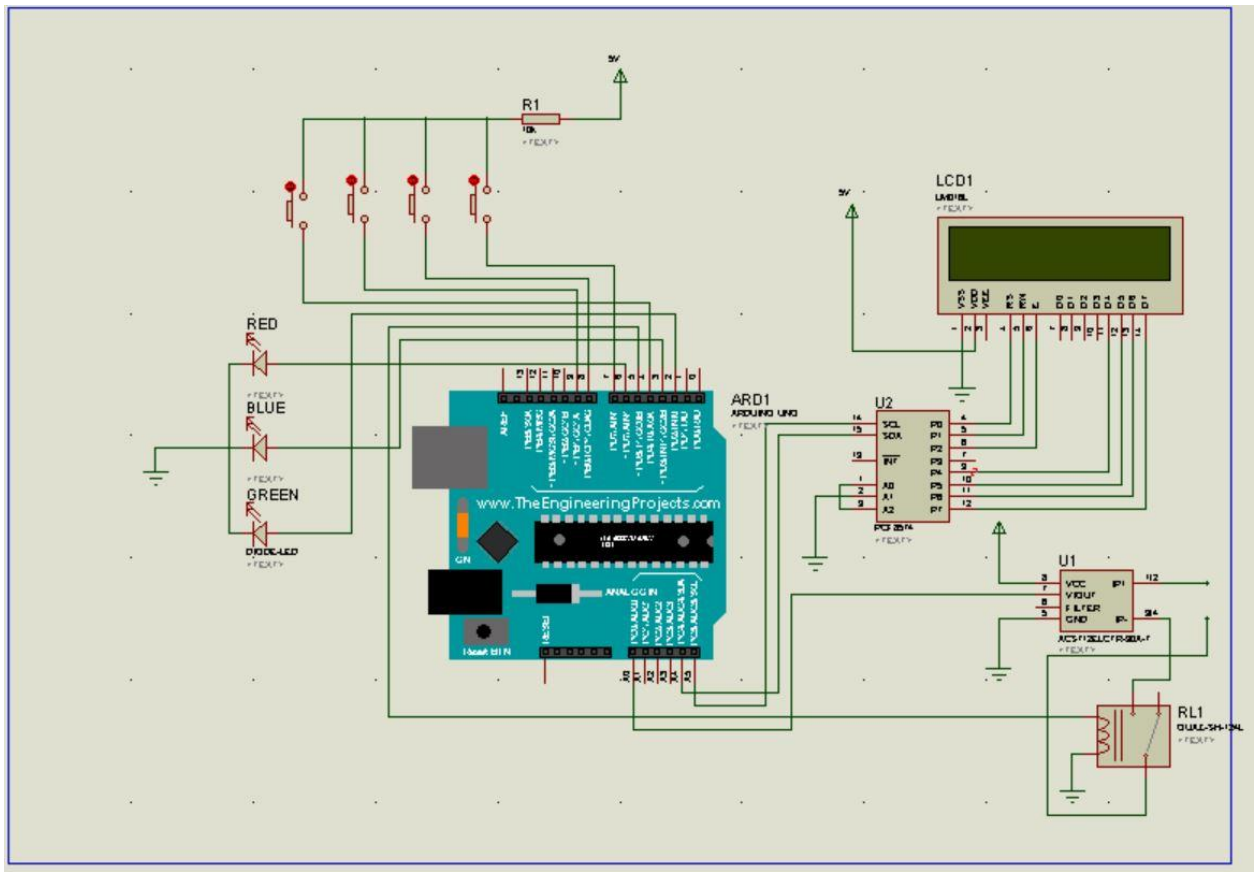
Picture5: Breadboard

WORKING PRINCIPLE

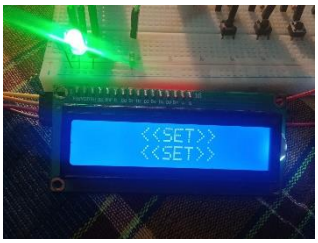
When we are going to use our module for first time. Firstly, a voltage of 5 volts have to be given to the electronic modules used in our system and grounding should be grounded. Now, we have to set our pick up current and loads. Before setting pick up current we also have to know our load capacity. For Setup our pickup current value, we have to input our value by using push button. In our module, here is 4 input push button, one of is increasing button and one of is decreasing button and other two is set up and reset button. If we need to increase our value, we have to push increasing button and if we have how to decrease or value, we have to push decreasing button. Every increasing push button or decreasing push button push will increase or decrease 0.25 value. After inputting pick up current, we have to push the set up button. Now our module is set for overload protection.

let's say, we set our pickup current to 0.25. After setting pick up current our microcontroller will continuously check and comparing the current by using current sensor for if the present current is over our pickup current or not? when our microcontroller found overcurrent comparing to our set pickup current, our LCD display will show fault detection and wait for 3 second. In this 3 second, our microprocessor will continuously check the current if the current value or overload is going to normal or not? If the value of current or overload is not gone normal, then the microcontroller will send signal to relay to trip the relay and protect our load or circuit. If the value of current or overload returns to normal within 3 seconds, then the system will continue to its normal state again. That's how our Microcontroller or electronic based overload protection works.

Circuit Diagram:



Display:



Picture1: Pickup current is set.

Picture3: Showing Pickup current and instant current value.

Picture2: The system has been reset.

Picture4: Fault detected and waiting for 3 seconds.

Code:

```
#include <LiquidCrystal_I2C.h>
#include "ACS712.h"
#include <IRremote.h>
#include<EEPROM.h>
ACS712 sensor(ACS712_05B, A0);
//ACS712_05B
//ACS712_20A for 20 Amp type
//ACS712_30A for 30 Amp type
float I = 0;
float cc = 0;
float data = 0;
int pp = 0;
int pin = 7;
int pin2 = 8;
int pin3 = 9;
int pin4 = 4;
int ccc = 0;
int bbb = 0;
int hh = 0;
int self = 0;
float picup_current = 0; // intial cosidering current for controlling
LiquidCrystal_I2C lcd(0x27, 16, 2); // set the LCD address to 0x3F for a 16 chars and 2 line display
void setup() {
digitalWrite(10, HIGH);
lcd.init(); // lcd display initialize
lcd.clear();// display clearing
lcd.backlight();    // Make sure backlight is o
pinMode(pin, INPUT); //  "
pinMode(pin2, INPUT); // "
pinMode(pin3, INPUT); // "
pinMode(pin4, INPUT); // "
pinMode(2, OUTPUT);// green indecator
pinMode(3, OUTPUT); // blue indecator
pinMode(6, OUTPUT); // red indecator
pinMode(5, OUTPUT); // relay coil
```

```
pinMode(10, OUTPUT); //rst
digitalWrite(2, LOW);
digitalWrite(3, HIGH);
digitalWrite(6, LOW);
digitalWrite(5, HIGH);
sensor.calibrate();
Serial.begin(9600);
delay(500);

}

void loop() {

unsigned long sec = millis();
sec = sec / 1000;
int input = digitalRead(pin); //++
int input2 = digitalRead(pin2); // --
int input3 = digitalRead(pin3); // reset
int input4 = digitalRead(pin4); // set
I = sensor.getCurrentAC();
//ignoring the value below 0.09
if (I < 0.09) {
I = 0;
}
lcd.setCursor(0, 1);
lcd.print("Current >");
lcd.print(I);
lcd.print("A");
//Serial.println(I);
lcd.setCursor(0, 0);
lcd.print("Pickup C>");
lcd.print(cc);
lcd.print("A");

// "Increasing" section
if (input == 1 && ccc == 0 )
{
cc = cc + 0.25;
cc = cc;
ccc = !ccc;

}

else if (input == 0 && ccc == 1 )
{
ccc = 0;
}

// "Decreasing" section
else if (input2 == 1 && bbb == 0 && cc > 0)
{
cc = cc - 0.25;
bbb = !bbb;
}

else if (input2 == 0 && bbb == 1 )
{
bbb = 0;
}

// "Reset" section
else if (input3 == 1 &&  hh > 0)
```

```
{
digitalWrite(5, HIGH); // relay off
digitalWrite(6, HIGH); // red led on
digitalWrite(2, LOW);
digitalWrite(3, LOW);
picup_current = cc;
lcd.clear();
lcd.setCursor(4, 0);
lcd.print("<<RESET>>");
lcd.setCursor(4, 1);
lcd.print("<<RESET>>");
delay(1000);
digitalWrite(10, LOW);
hh = 0;
cc = 0;
}

// auto reseting and 3 sec  time delay section
else if (I > picup_current && hh > 0)
{
lcd.clear();
lcd.setCursor(0, 0);
lcd.print("Fault Detect");
lcd.setCursor(0, 1);
lcd.print("Wait 3 SEC");
//Serial.print(I);
delay(3000);
check();

}

// "SET" section

else if (input4 == 1 && cc > 0 )
{
picup_current = cc;
float fg = map(picup_current * 100, 0, 500, 0, 255);
EEPROM.write('L1', fg);
digitalWrite(5, LOW); // relay on
digitalWrite(3, LOW);
digitalWrite(6, LOW);
digitalWrite(2, HIGH ); // green led  on
lcd.clear();
lcd.setCursor(5, 0);
lcd.print("<<SET>>");
lcd.setCursor(5, 1);
lcd.print("<<SET>>");
delay(1000);
hh = 5;
}

// Self Decission section for set
else if ( sec >= 20 && sec <= 30 && hh == 0 && self == 0 )
{
data = EEPROM.read('L1');
Serial.println(data);
data = map(data, 0, 255, 0, 500);
cc = data / 100;
picup_current = cc + 0.20;
digitalWrite(5, LOW); // relay on
digitalWrite(3, LOW);
digitalWrite(6, LOW);
```

```
digitalWrite(2, HIGH ); // green led  on
lcd.clear();
lcd.setCursor(5, 0);
lcd.print("<<SET>>");
lcd.setCursor(5, 1);
lcd.print("<<SET>>");
delay(1000);
self = 5;
hh = 5;
}
else
{
}
}
// Stability checking  section after 3 sec
void check()
{
I = sensor.getCurrentAC();
if (I > picup_current)
{
//Serial.print("YES");
digitalWrite(5, HIGH); // relay off
digitalWrite(6, HIGH); // red led on
digitalWrite(2, LOW);
digitalWrite(3, LOW);
picup_current = cc;
lcd.clear();
lcd.setCursor(4, 0);
lcd.print("<<RESET>>");
lcd.setCursor(4, 1);
lcd.print("<<RESET>>");
delay(1000);
digitalWrite(10, LOW);
hh = 0;
cc = 0;
}
else
{
Serial.print("NO");
}
}
```

- **Project Reference and file Link : https://github.com/Rafsan12345/Induestrial-Control/tree/main/Induestrial%20control/Overload_Protector**

- **Scan the QR code to know details about the project**