EMERGANCY 132kV BAY CONTROLLER UNIT AT MATUGAMA GS

OFFICE OF CHIEF ENGINEER - TRANSMISSION OPERATION AND MAINTAINANCE SOURTHERN REGION

03th February 2023

CONTENTS

Chapter	Title	Page
1	**Introduction**	**1**
	1.1 Introduction to 132kV Bay Controllers	1
	1.2 Existing 132kV Bay Controllers in Matugama GS	1
	1.3 Control Logics in Existing 132kV Line Bay Controller	4
	1.4 Concerns of Existing 132kV Bay Controllers in Matugama GS	4
	1.5 Introduction to Arduino Technology	4
2	**Control Function of Emergency BCU**	**6**
	2.1 Digital Inputs to Emergency BCU	6
	2.2 Analog Inputs to Emergency BCU	10
	2.3 Outputs from Emergency BCU	11
	2.4 Flow of Single Operation	13
	2.5 Busbar 01 Disconnector Operations	14
	2.6 Busbar 02 Disconnector Operations	14
	2.7 Line Disconnector Operations	15
	2.8 Earth Switch Operations	15
	2.9 External Syc-Check relay	16
	2.10 Circuit Breaker Operations	18
	2.11 Alarms from Control Function of Emergency BCU	18
3	**Event Recording Function of Emergency BCU**	**20**
	3.1 Event Recording Algorithm	20
4	**Network Connection of Emergency BCU**	**24**
	4.1 Introduction to Network Connection	24
	4.2 User Interface	26
	4.3 Network Connection Code Explanations	27
5	**Testing of Emergency BCU**	**28**
	5.1 Introduction to 132kV Circuit Breaker Operation Testing	28
	5.2 Switching Release Testing	28
	5.3 Synch/Energizing Release Check	30
	5.4 Close command with Close command time testing.	31
	5.5 Introduction to 132kV Busbar 1 and 2 Isolator Testing	32
	5.6 Operation Testing of 132kV Busbar 1 Isolator	32

5.7 Operation Testing of 132kV Busbar 2 Isolator 32
5.8 Introduction to 132kV Line Isolator and Earth-switch Testing 33
5.9 Operation Testing of 132kV Earth-Switchk 33

Annexures
1.Drawings
2.Codings

1.1 Introduction to 132kV Bay Controllers

The main function of BCU is to monitor the whole system remotely. The status of primary equipment or auxiliary devices can be obtained from auxiliary contacts. Therefore it is possible to detect and in- 1 -dicate both the OPEN and CLOSED position or a fault or intermediate circuit-breaker or auxiliary contact position.

In addition to the monitoring functions, BCU also supports that are required for operating medium-voltage or high-voltage substations. The main application is reliable control of switching and other processes.

With integrated logic, the user can set, via specific functions for the automation of switchgear or substation. Functions are activated via function keys, binary input, or communication interface.

1.2 Existing 132kV Bay Controllers in Matugama GS

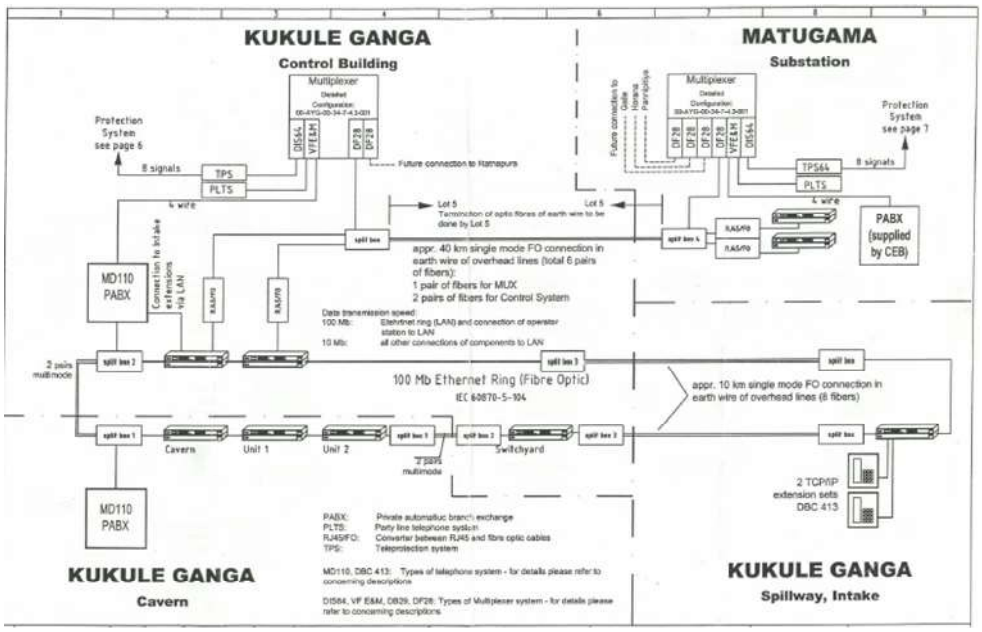


Figure1.1 – Fiber Optic Ethernet Ring in Kukule PS and Matugama GS

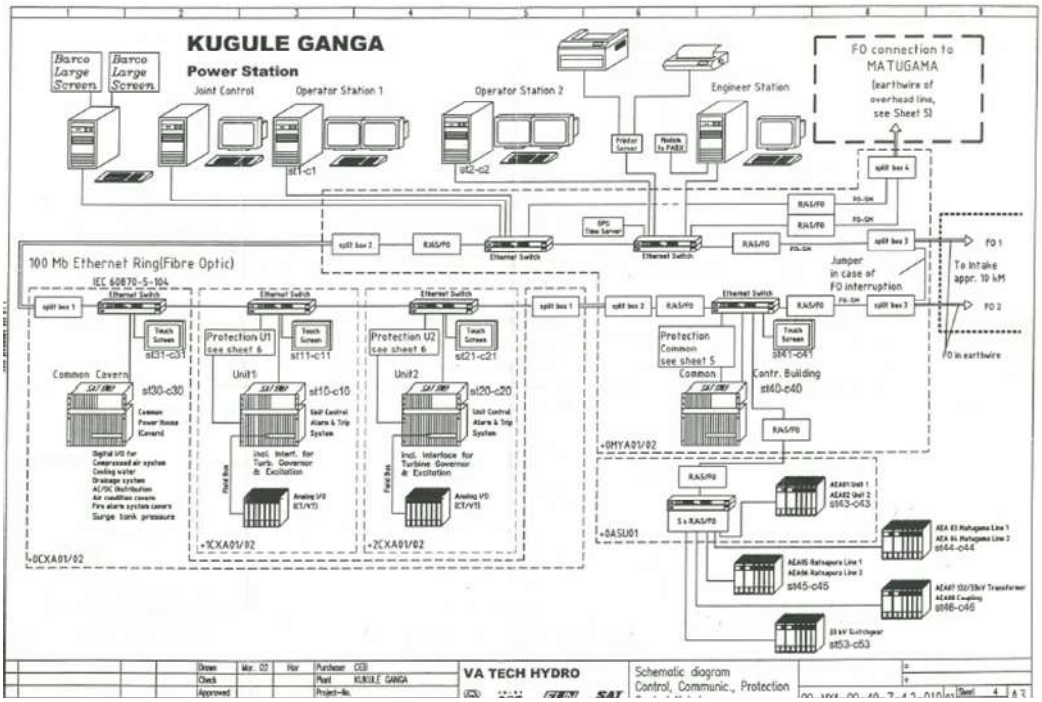


Figure1.2 – SAS Architecture in Kukule PS and Matugama GS Part 1

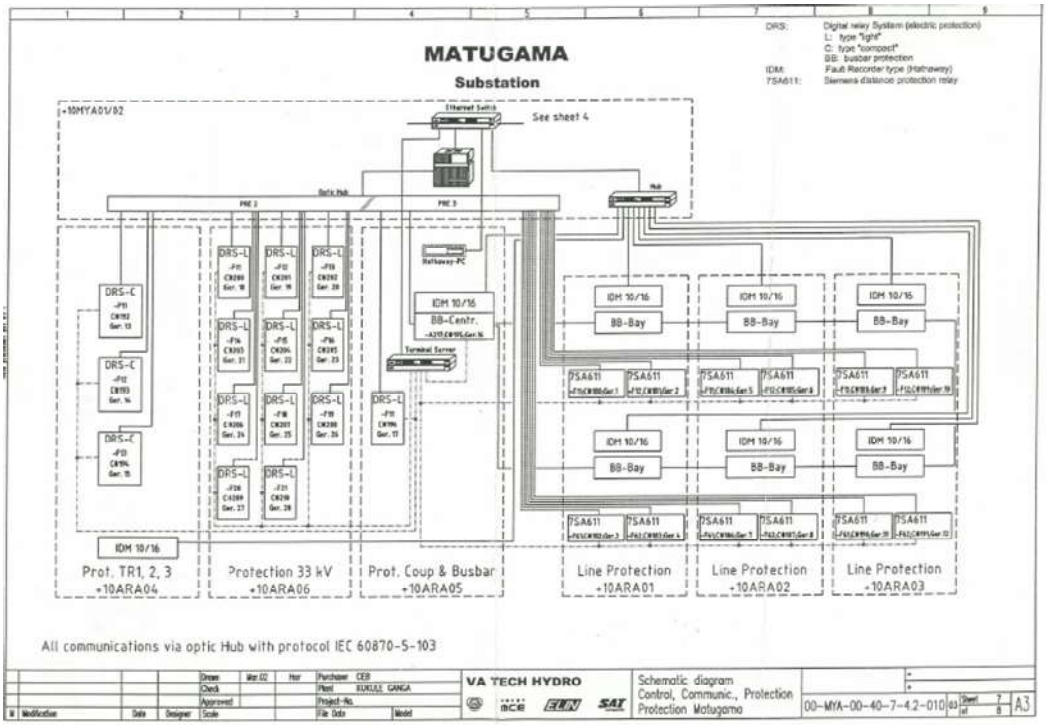


Figure1.3 – SAS Architecture in Kukule PS and Matugama GS Part 2

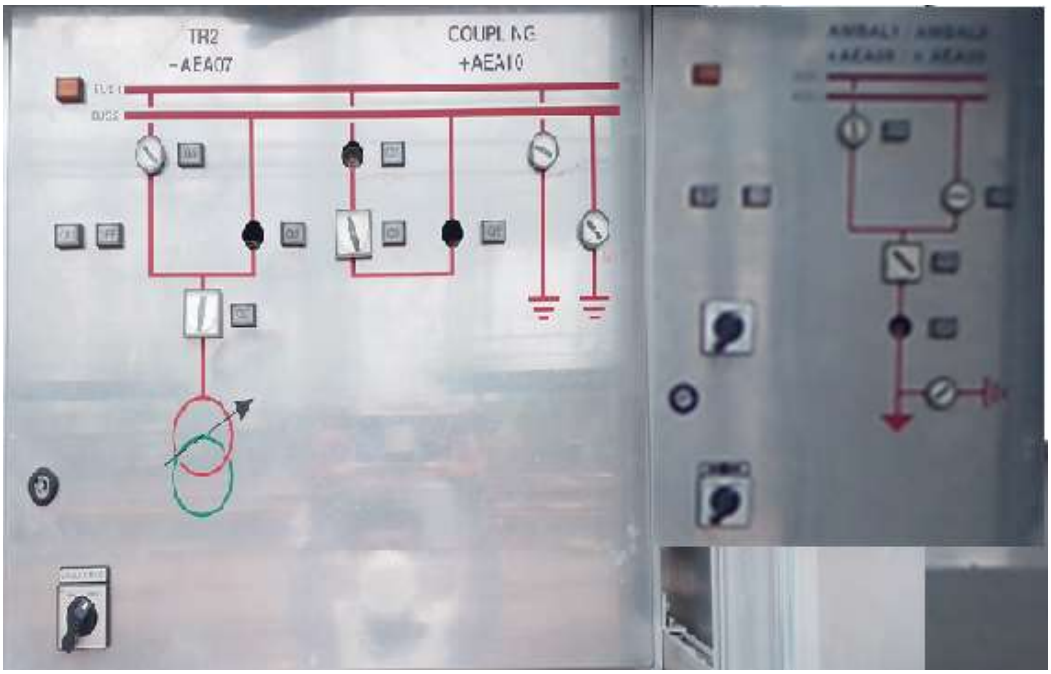


Figure1.4 – Mimic of 132kV Tr. Bay, Buscoupler Bay and Line Bay

1.3 Control Logics in Existing 132kV Line Bay Controller

Control Logics in Existing 132kV Line Bay Controllers are in Kukule Ganga Hydropower Project Lot 4.2 : Transformers and Switching Equipment drawing no:-10-AHA-00-41-7-4.2-232 and those are attached as Annex 01.

1.4 Concerns of Existing 132kV Bay Controllers in Matugama GS

Existing 132kV Bay Controllers in Matugama GS were replaced several times by spare controllers. Now spare controllers for Matugama GS are ended due to the above replacements.

Procurement of spare controllers for Matugama GS was initiated by Tr. Assert Management at the beginning of 2022 and now it was halted due to controller manufacture issue.

Matugama GS is one of the most important grid substations belonging to CEB. It connects Kukule Ganga Power Station to the main 132kV grid. Therefore it is very much required to find a good alternative solution for these controller relays.

1.5 Introduction to Arduino Technology

Arduino is an open-source hardware and software company, project, and user community that designs and manufactures single-board microcontrollers and microcontroller kits for building digital devices. Its hardware products are licensed under a CC BY-SA license, while the software is licensed under the GNU Lesser General Public License (LGPL) or the GNU General Public License (GPL), permitting the manufacture of Arduino boards and software distribution by anyone. Arduino boards are available commercially from the official website or through authorized distributors.

Arduino board designs use a variety of microprocessors and controllers. The boards are equipped with sets of digital and analog input/output (I/O) pins that may be interfaced to various expansion boards ('shields') breadboards (for prototyping) and other circuits. The boards feature serial communications interfaces, including Universal Serial Bus (USB) on some models, which are also used for loading programs. The microcontrollers can be programmed using the C and C++ programming languages, using a standard API which is also known as the Arduino language, inspired by the Processing language and used with a modified version of the Processing IDE.

The Arduino project began in 2005 as a tool for students at the Interaction Design Institute in Ivrea, Italy, aiming to provide a low-cost and easy way for novices and professionals to create devices that interact with their environment using sensors and actuators. Common examples of such devices intended for beginner hobbyists include simple robots, thermostats, and motion detectors.

2.1 Digital Inputs to Emergency BCU

Digital inputs to the controller are taken through a small auxiliary relay as shown in Figure 2.1. Those auxiliary relays are powered through auxiliary contacts of relevant equipment.

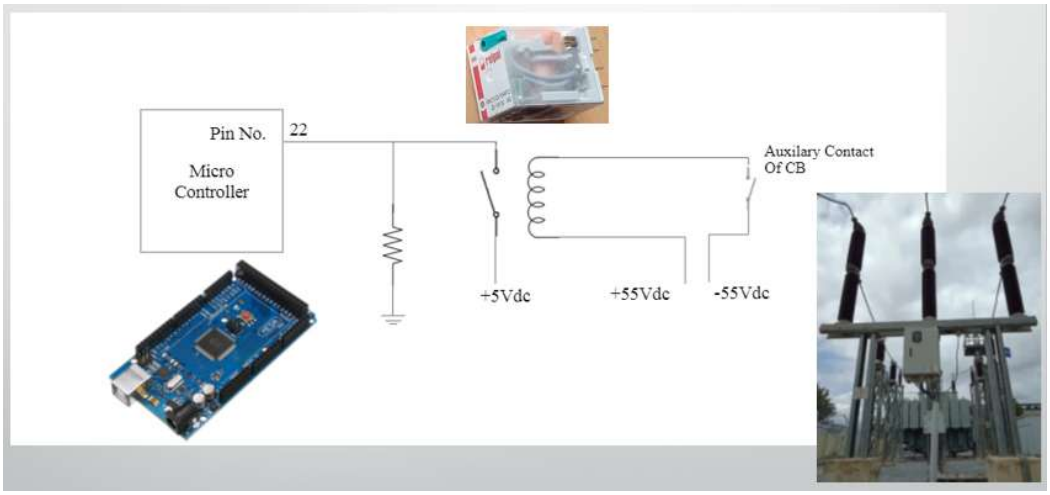


Figure 2.1 – Digital Inputs to Controller

Digital input stating the state of the Circuit breaker ON position is fed into Digital input No.22. All the other inputs are fed as per the following table numbers 2.1 and 2.2.

Equipment	Description	Status	Arduino Inputs
Circuit Breaker	CB_ON_IN	ON	22
	CB_OFF_IN	OFF	24
BUS D/S 01	B_DS01_ON_IN	ON	26
	B_DS01_OFF_IN	OFF	28
BUS D/S 02	B_DS02_ON_IN	ON	30
	B_DS02_OFF_IN	OFF	32
LINE D/S	L_DS_ON_IN	ON	34
	L_DS_OFF_IN	OFF	36
Earth S/W	ES_ON_IN	ON	38
	ES_OFF_IN	OFF	40
BUS BAR EARTH S/W	ES_BB1_ON_IN	ON	42
	ES_BB1_OFF_IN	OFF	44

Table 2.1 – Digital Inputs to Arduino Mega Controller-1

Equipment	Description	Status	Arduino Inputs
BUS BAR EARTH S/W	ES_BB1_ON_IN	ON	46
	ES_BB1_OFF_IN	OFF	48
SYNC	FEEDER_SYNC_IN		50
BUS COUPLER	BUS_COUP_ON_IN		52
VT MCB TRIP	VT_MCBTRIP_ON_IN		53
CB OFF ORDER	CB_OFF_ORDER_IN		45
CB ON ORDER	CB_ON_ORDER_IN		43
SYNC	FEEDER_SYNC_IN_CMD		17
BUS VT MCB TRIP	VT_MCBTRIP_ON_CMD		16
LINE VT R-PHASE	Line_Voltage		A0
BB1 VT MCB TRIP	BB1_ VT_MCBTRIP_ON_CMD		
BB2 MCB TRIP	BB2_ VT_MCBTRIP_ON_CMD		

Table 2.2 – Digital Inputs to Arduino Mega Controller-2

The following figure shows the way of assigning names for pins of the Arduino mega controller.

```
//Defining inputs 22 to 53**********
const int CB_ON_IN              = 22;
const int CB_OFF_IN             = 24;
const int B_DS01_ON_IN          = 26;
const int B_DS01_OFF_IN         = 28;
const int B_DS02_ON_IN          = 30;
const int B_DS02_OFF_IN         = 32;
const int L_DS_ON_IN            = 34;
const int L_DS_OFF_IN           = 36;
const int ES_ON_IN              = 38;
const int ES_OFF_IN             = 40;
const int ES_BB1_ON_IN          = 42;
const int ES_BB1_OFF_IN         = 44;
const int ES_BB2_ON_IN          = 46;
const int ES_BB2_OFF_IN         = 48;
const int FEEDER_SYNC_IN        = 50;
const int BUS_COUP_ON_IN        = 52;
const int VT_MCBTRIP_ON_IN      = 53;
const int CB_OFF_ORDER_IN       = 45;
const int CB_ON_ORDER_IN        = 43;
```

Figure 2.2 – Assigning names to digital pins

Above assigned pins shall be configured as digital inputs as given in Figure 2.3.

```
//Assigning inputs to digital IO******
pinMode(CB_ON_IN, INPUT); // assigning
pinMode(CB_OFF_IN, INPUT);
pinMode(B_DS01_ON_IN, INPUT);
pinMode(B_DS01_OFF_IN, INPUT);
pinMode(B_DS02_ON_IN, INPUT);
pinMode(B_DS02_OFF_IN, INPUT);
pinMode(L_DS_ON_IN, INPUT);
pinMode(L_DS_OFF_IN, INPUT);
pinMode(ES_ON_IN, INPUT);
pinMode(ES_OFF_IN, INPUT);
pinMode(ES_BB1_ON_IN, INPUT);
pinMode(ES_BB1_OFF_IN, INPUT);
pinMode(ES_BB2_ON_IN, INPUT);
pinMode(ES_BB2_OFF_IN, INPUT);
pinMode(BUS_COUP_ON_IN, INPUT);
pinMode(VT_MCBTRIP_ON_IN, INPUT);
pinMode(CB_ON_ORDER_IN, INPUT);
pinMode(CB_OFF_ORDER_IN, INPUT);
pinMode(FEEDER_SYNC_IN, INPUT);
```

Figure 2.3 – Configuring digital pins as Inputs

2.2 Analog Inputs to Emergency BCU

Analog inputs are taken to the controller as shown in Figure 2.4.

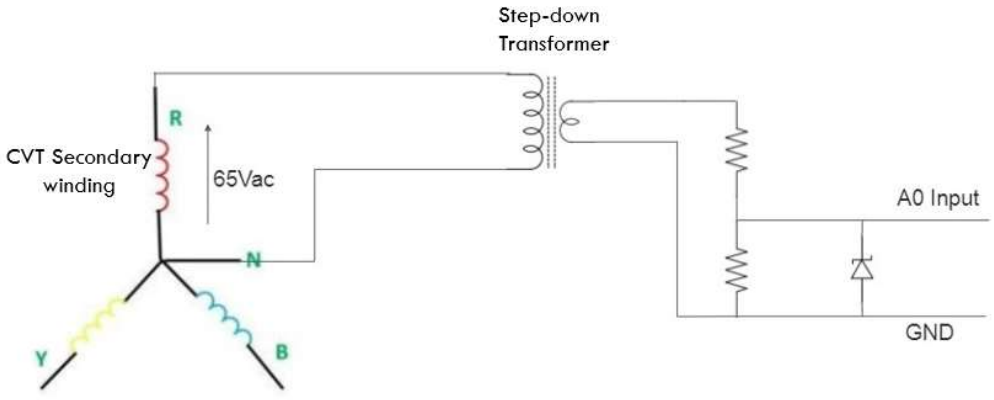


Figure 2.4 – Analog Inputs to Controller.

2.3 Outputs from Emergency BCU

Outputs from Micro Controller is taken through auxiliary relays which are capable of handling 110V DC. Opto-couplers are utilized to isolate the microcontroller system from 110V DC system.

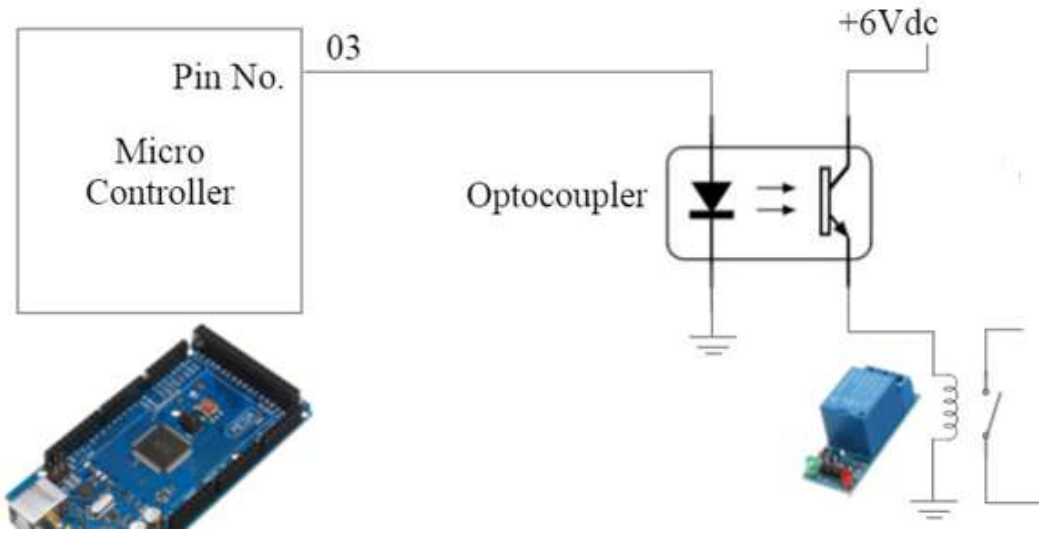


Figure 2.5 – Output Relay Driving Circuit

Equipment	Description	Status	Arduino Output
CB ON	CB_ON_CMD		14
CB PRE ON	CB_PRE_ON_CMD		2
BUS DS 01	BUS01_DSQ01_RELEASE_ON_CMD	ON	3
	BUS01_DSQ01_RELEASE_OFF_CMD	OFF	4
BUS DS 02	BUS02_DSQ02_RELEASE_ON_CMD	ON	5
	BUS02_DSQ02_RELEASE_OFF_CMD	OFF	6
LINE DS	L_DS_RELEASE_ON_CMD	ON	19
	L_DS_RELEASE_OFF_CMD	OFF	20
E. SW	ES_RELEASE_ON_CMD		18

Table 2.3 – Digital Outputs from Arduino Mega Controller

```
//Defining outputs 01 to 19 **********************
const int CB_ON_CMD                     = 14;
const int CB_PRE_ON_CMD                 = 2;
const int BUS01_DSQ01_RELEASE_ON_CMD    = 3;
const int BUS01_DSQ01_RELEASE_OFF_CMD   = 4;
const int BUS02_DSQ02_RELEASE_ON_CMD    = 5;
const int BUS02_DSQ02_RELEASE_OFF_CMD   = 6;
const int L_DS_RELEASE_ON_CMD           = 19;
const int L_DS_RELEASE_OFF_CMD          = 20;
const int ES_RELEASE_ON_CMD             = 18;
```

```
//Assigning outputs to digital IO*************
pinMode(CB_ON_CMD, OUTPUT);
pinMode(CB_PRE_ON_CMD, OUTPUT);
pinMode(BUS01_DSQ01_RELEASE_ON_CMD, OUTPUT);
pinMode(BUS01_DSQ01_RELEASE_OFF_CMD, OUTPUT);
pinMode(BUS02_DSQ02_RELEASE_ON_CMD, OUTPUT);
pinMode(BUS02_DSQ02_RELEASE_OFF_CMD, OUTPUT);
pinMode(L_DS_RELEASE_ON_CMD, OUTPUT);
pinMode(L_DS_RELEASE_OFF_CMD, OUTPUT);
pinMode(ES_RELEASE_ON_CMD, OUTPUT);
```

Figure 2.6 – Cording for Defining Digital Pins as Outputs

2.4 Flow of Single Operation

Following flow chart represent the flow of single operation.

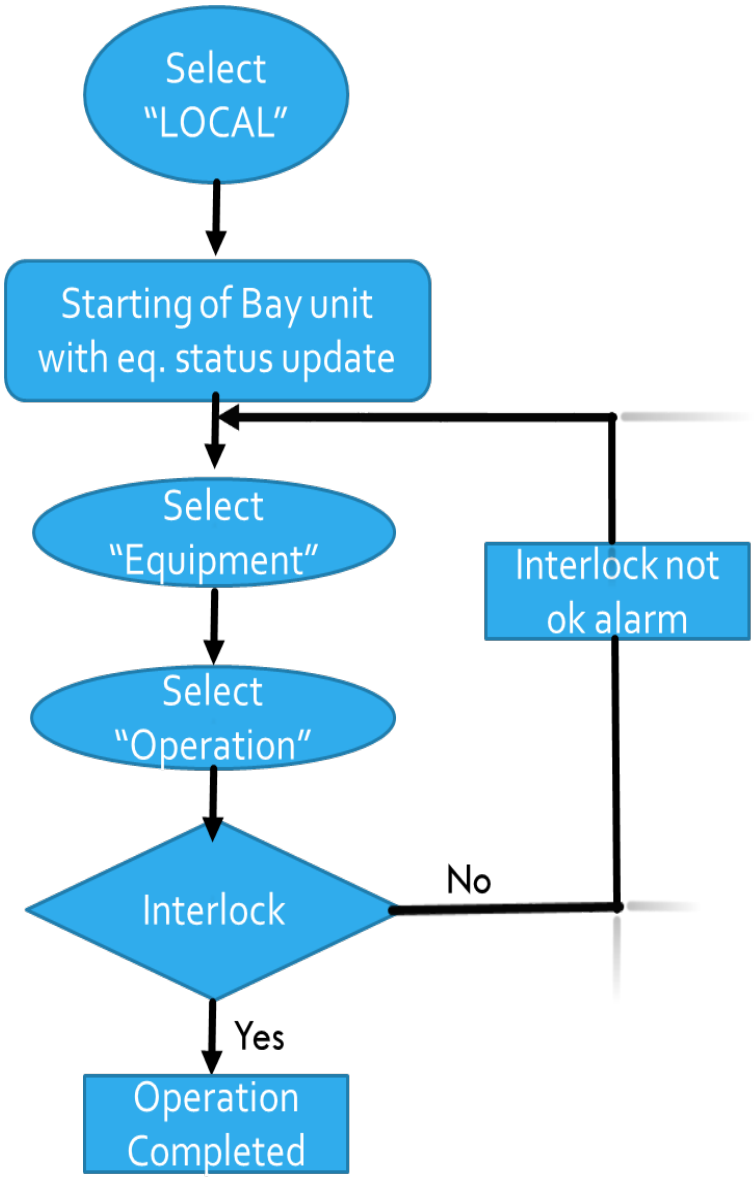


Figure 2.7 – Flow of Single Operation

As per the following figure, equipment select and ON or OFF operation buttons shall be pressed simultaneously for equipment command. However, it is required to energize the relay from the microcontroller as per the interlock conditions for successful equipment operation. Otherwise, it will give an “Interlock Not OK” alarm.

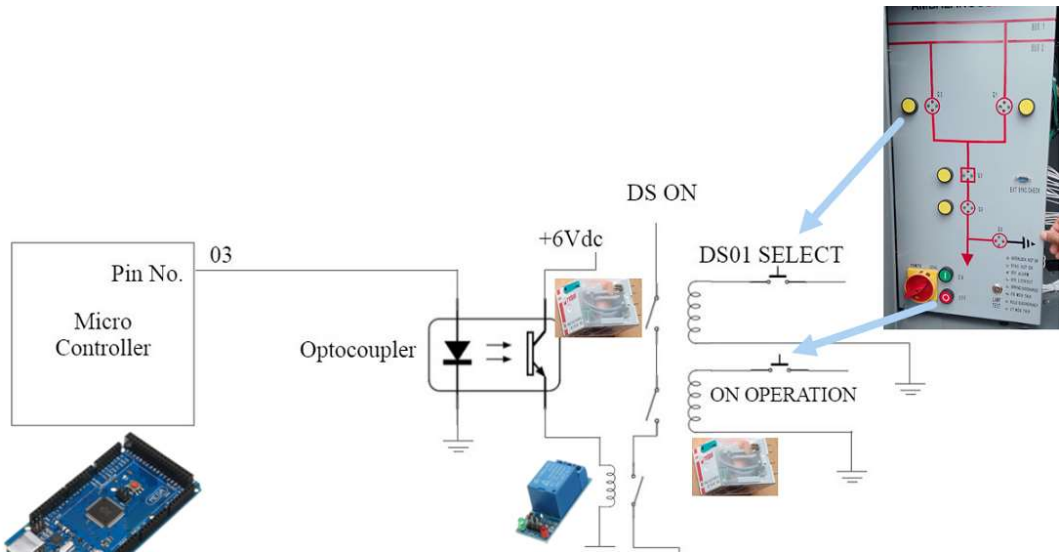


Figure 2.8 –Operation Path of Disconnector Switch

2.5 Busbar 01 Disconnector Operations

Control Logics for Busbar 01 disconnector was taken from existing drawings of Kukule Ganga Hydropower Project Lot 4.2: Transformers and Switching Equipment drawing no:-10-AHA-00-41-7-4.2-232 and those are attached as Annex 01. Those logics are coded as below (Figure 2.9).

```
//Q1 Open/Close Release Operation*********************************************************************

if  (((B_DS02_ON) && not(B_DS02_OFF) && not(ES_BB1_ON) && (ES_BB1_OFF) && (BUS_COUP_ON))
or ( (CB_OFF) && not(CB_ON)&&(B_DS02_OFF) && not (B_DS02_ON ) && not (ES_BB1_ON) && (ES_BB1_OFF))){

  digitalWrite(BUS01_DSQ01_RELEASE_ON_CMD,LOW);
  digitalWrite(BUS01_DSQ01_RELEASE_OFF_CMD,LOW);
}
else{digitalWrite(BUS01_DSQ01_RELEASE_ON_CMD,HIGH);
     digitalWrite(BUS01_DSQ01_RELEASE_OFF_CMD,HIGH);}
```

Figure 2.9 –Code of Bus 01 Disconnector Switching Release

2.6 Busbar 02 Disconnector Operations

Control Logics for Busbar 02 disconnector was taken from existing drawings of Kukule Ganga Hydropower Project Lot 4.2 : Transformers and Switching Equipment

drawing no:-10-AHA-00-41-7-4.2-232 and those are attached as Annex 01. Those logics are coded as below (Figure 2.10).

```
//Q2 Open/Close Release Operation*********************************************************************

if  (((B_DS01_ON) && not(B_DS01_OFF) && not(ES_BB2_ON) && (ES_BB2_OFF) && (BUS_COUP_ON))
or ((CB_OFF) && not(CB_ON) && (B_DS01_OFF) && not (B_DS01_ON ) && not (ES_BB2_ON) && (ES_BB2_OFF))){

  digitalWrite(BUS02_DSQ02_RELEASE_ON_CMD,LOW);
  digitalWrite(BUS02_DSQ02_RELEASE_OFF_CMD,LOW);
}
else{digitalWrite(BUS02_DSQ02_RELEASE_ON_CMD,HIGH);
     digitalWrite(BUS02_DSQ02_RELEASE_OFF_CMD,HIGH);}
```

Figure 2.10 –Code of Bus 02 Disconnector Switching Release

2.7 Line Disconnector Operations

Control Logics for Line disconnector was taken from existing drawings of Kukule Ganga Hydropower Project Lot 4.2: Transformers and Switching Equipment drawing no:-10-AHA-00-41-7-4.2-232 and those are attached as Annex 01. Those logics are coded as below (Figure 2.11).

```
//Q9 Release (L  D/S)**********************************

if ((CB_OFF) && not(CB_ON) && not(ES_ON) && (ES_OFF)){
   digitalWrite(L_DS_RELEASE_ON_CMD,LOW);
   digitalWrite(L_DS_RELEASE_OFF_CMD,LOW);}
   else {
  digitalWrite(L_DS_RELEASE_ON_CMD, HIGH);
  digitalWrite(L_DS_RELEASE_OFF_CMD,HIGH);}
```

Figure 2.11 –Code of Line Disconnector Switching Release

2.8 Earth Switch Release

Control Logics for Earth Switch Release were modified with the comments given by the Transmission Control and Protection System Branch. A sync check relay module connection is required for this Earth Switch Release function. It will not give a release without connecting it to the panel. It will not give the Earth Switch Release until Line Voltage is reduced to a voltage less than 20% of its rated value even after connecting

the relay module. “No Line Voltage” binary input is taken from the Sync check relay module.

```
//Q8 - ES Release Operation************************************************

if   ( NO_LINE_VOL && L_DS_OFF && not(L_DS_ON) && not (VT_MCBTRIP_ON))
       {digitalWrite(ES_RELEASE_ON_CMD,LOW);}
else   {digitalWrite(ES_RELEASE_ON_CMD,HIGH);}
```

Figure 2.12 –Code for Earth Switch Release

2.9 External Sync-Check relay

External Sync-Check unit is developed based on Alstom KVAS 100 relay to synchronize the Line Voltage and Bus Voltage. This unit allows LL-LB, DL-LB, and DL-DB conditions by closing one relay contact. That contact will give input to the controller. Then controller changes the sync-ok condition from zero to one.

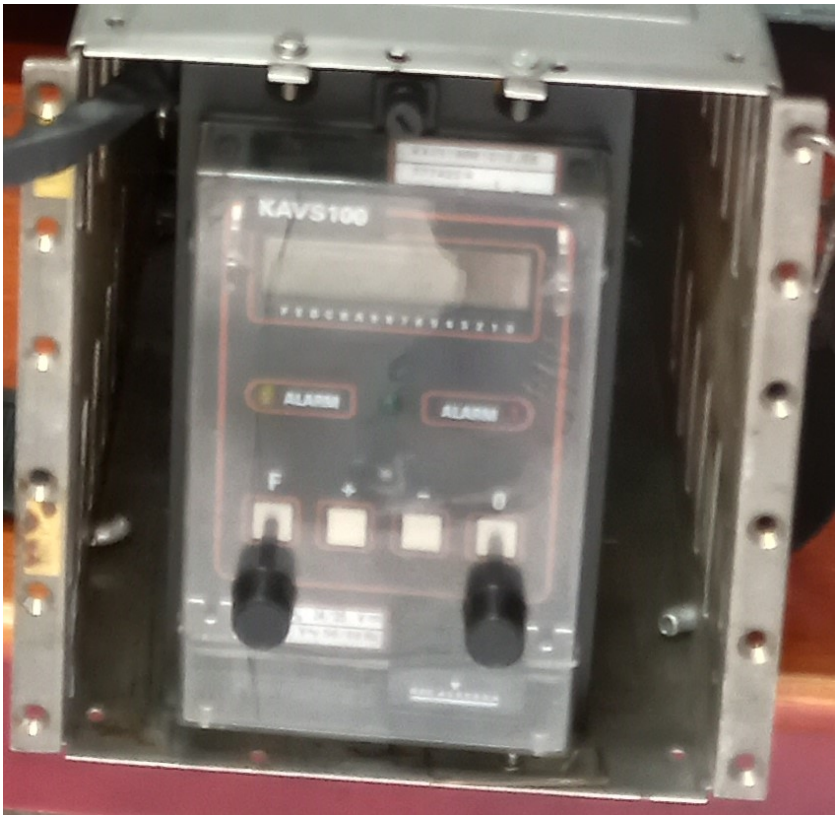


Figure 2.13 –Sync-check relay

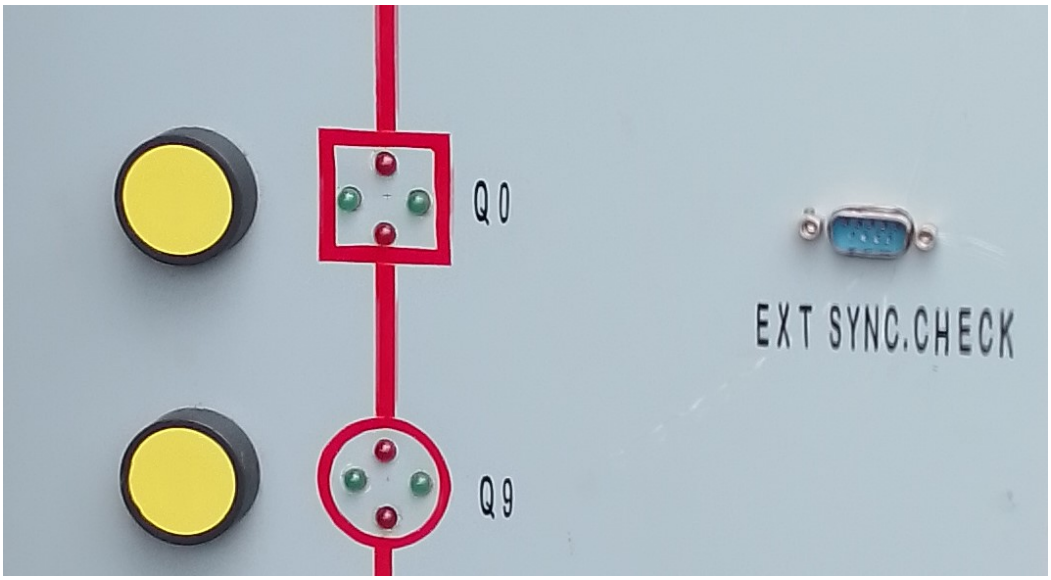


Figure 2.14 –Sync-check relay connection port

Figure 2.15 –Sync-check relay connection

2.10 Circuit Breaker Operations

Circuit Breaker operation is the most important operation in the bay. Circuit breaker operation logic is given in Figures 2.16 and 2.17.

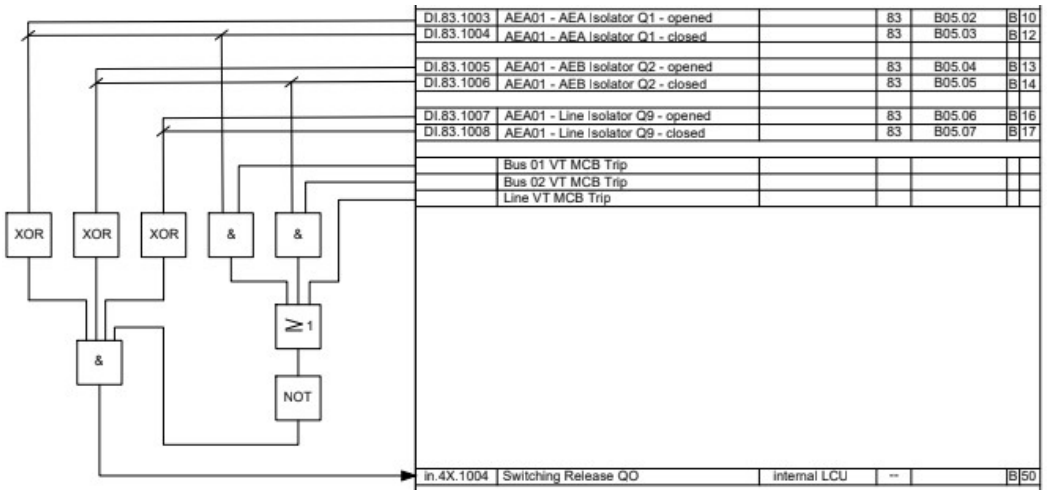


Figure 2.16 –Logic for Circuit Breaker Switching Release

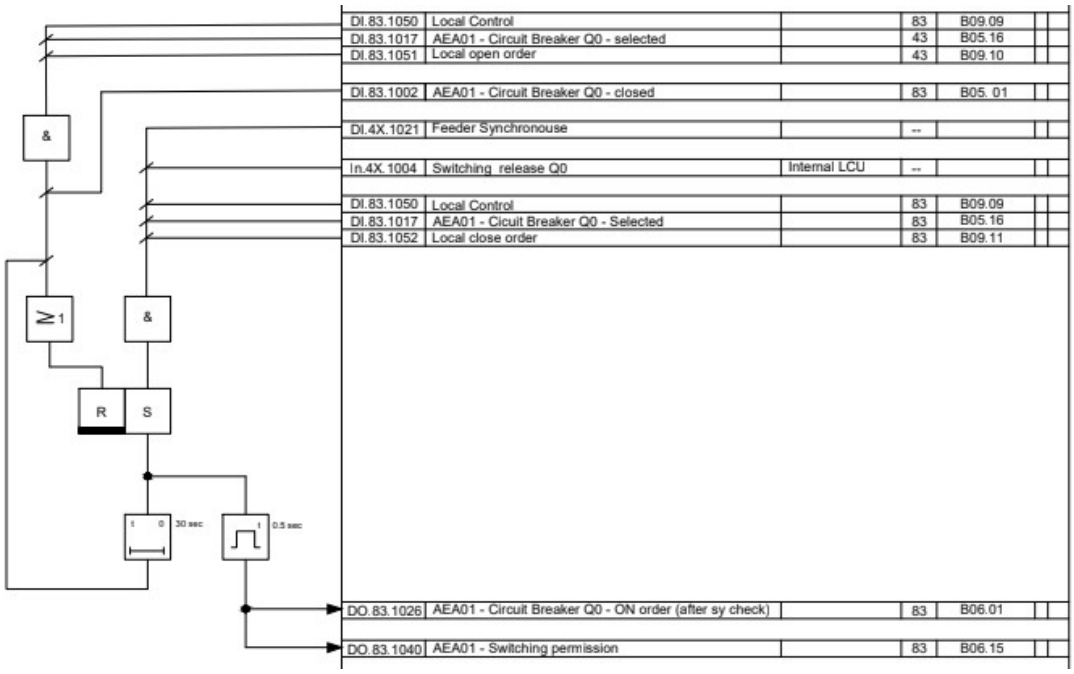


Figure 2.17 –Logic for Circuit Breaker Close

The coding for the above logic is given below in Figure 2.18. Several loops were utilized to create pulses and timers.

```
/// Final CB_ON_CMD
  if (  (CB_ON_ORDER or CB_ON_i) && DEAD_SYSTEM && CB_SWITCHING_RELEASE &&
          (not(PROT_TRIP))&& (not(CB_OFF_ORDER))&&(not(CB_ON_IN)) ) {
          CB_ON_i=1;
          CB_i++;

          if (CB_i<CB_CLS_TIME){digitalWrite(CB_ON_CMD,LOW);}
          else { digitalWrite(CB_ON_CMD,HIGH);}

          if (CB_CLS_WAIT_TIME<CB_i){CB_ON_i=0;}

  }else if (  (CB_ON_ORDER or CB_ON_i) && FEEDER_SYNC && CB_SWITCHING_RELEASE &&
            (not(PROT_TRIP))&& (not(CB_OFF_ORDER)&&(not(CB_ON_IN))) ) {
          CB_ON_i=1;
          CB_i++;

          if (CB_i<CB_CLS_TIME){digitalWrite(CB_ON_CMD,LOW);}
          else { digitalWrite(CB_ON_CMD,HIGH);}
          if (CB_CLS_WAIT_TIME<CB_i){CB_ON_i=0;}

  }else if (PROT_TRIP or CB_OFF_ORDER or CB_ON_IN) {
   CB_ON_i=0;
   CB_i=0;}

   else {
     digitalWrite(CB_ON_CMD,HIGH);
     CB_ON_i=0;
     CB_i=0;}
```

Figure 2.18 –Cording for Circuit Breaker Close

3.1 Event Recording Algorithm

The event recording function was requested from the Protection development branch and it was implemented as below. It utilized a real-time clock to take time stamps for events and an SD card writer to record those events.

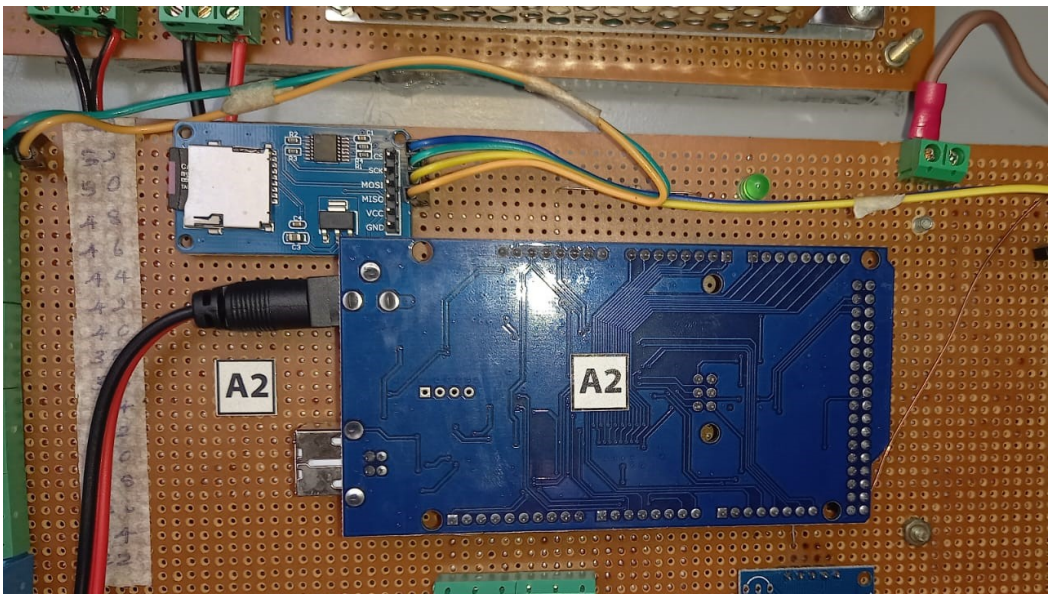


Figure 3.1 – Event Recorder

The location of the event recorder in the unit is marked in Figure 3.2.

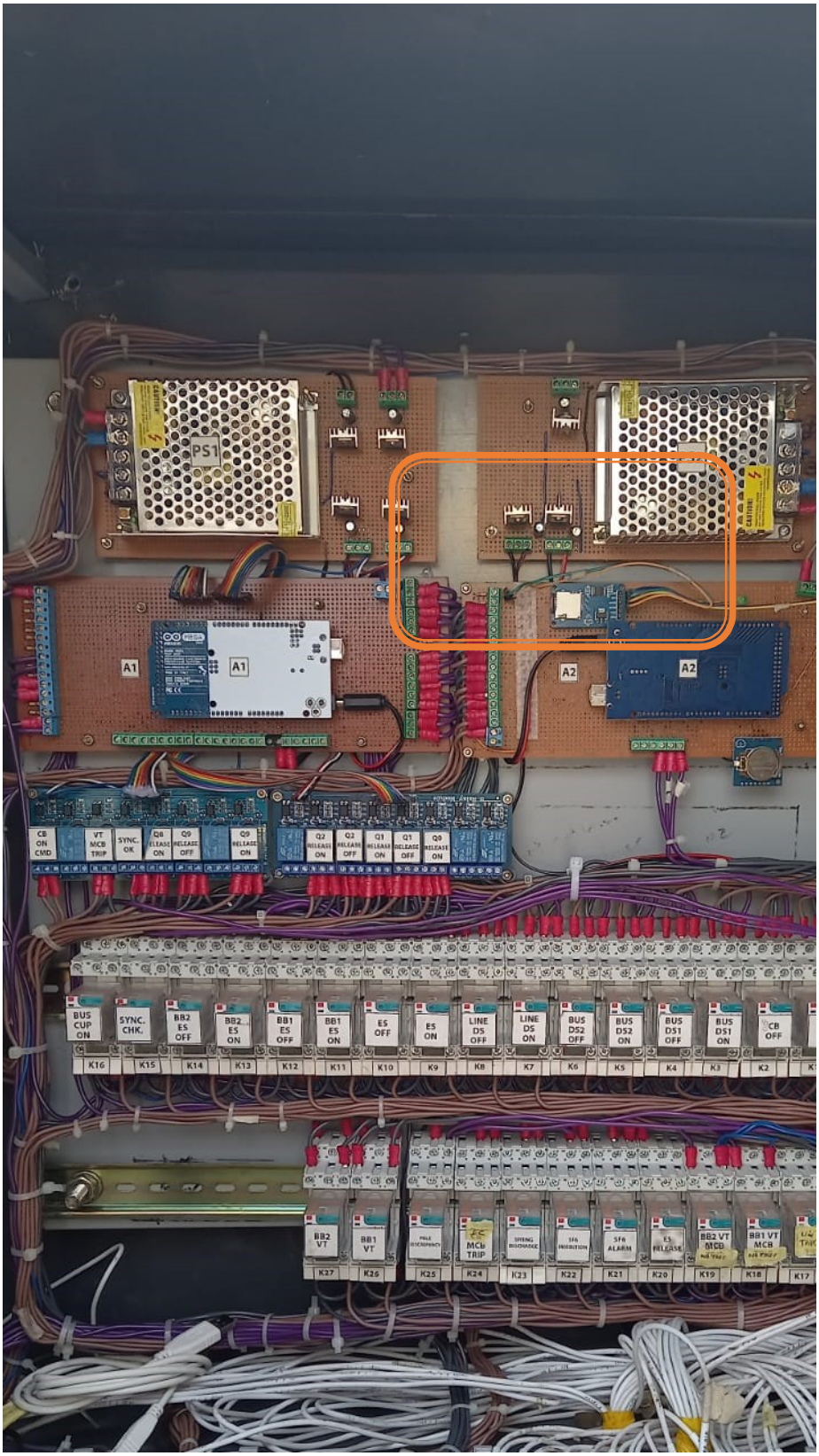


Figure 3.2 – Event Recorder Location

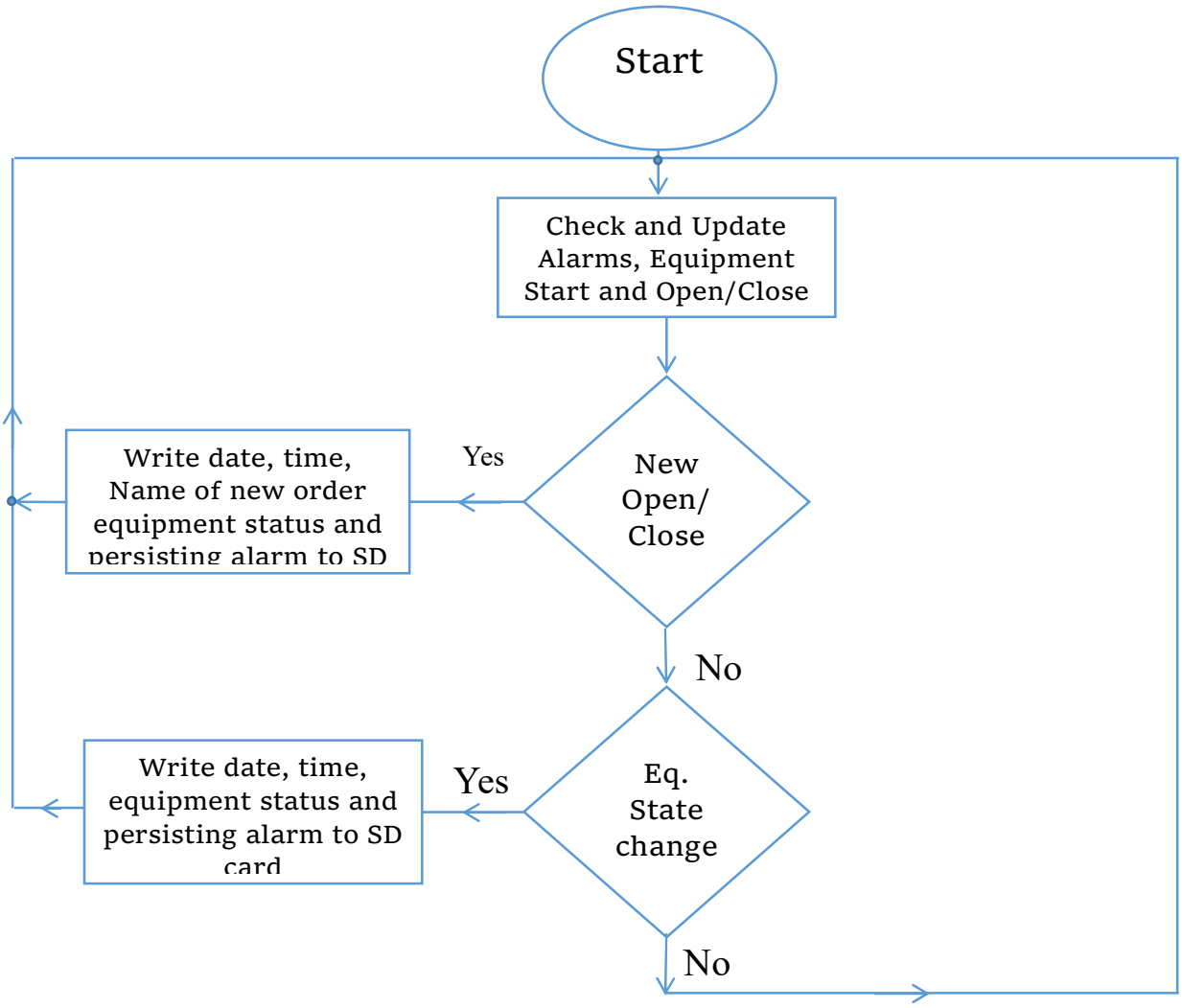


Figure 3.3 – Event Recorder Flow Chart

The flow chart of the event recorder program is shown in Figure 3.3. It starts after turning on the unit. The battery of the clock unit shall be maintained in good condition to take up-to-date time stamps for event records.

A screen shot of an event log file is shown in Figure 3.4. Event log files are saved in ".txt" format.

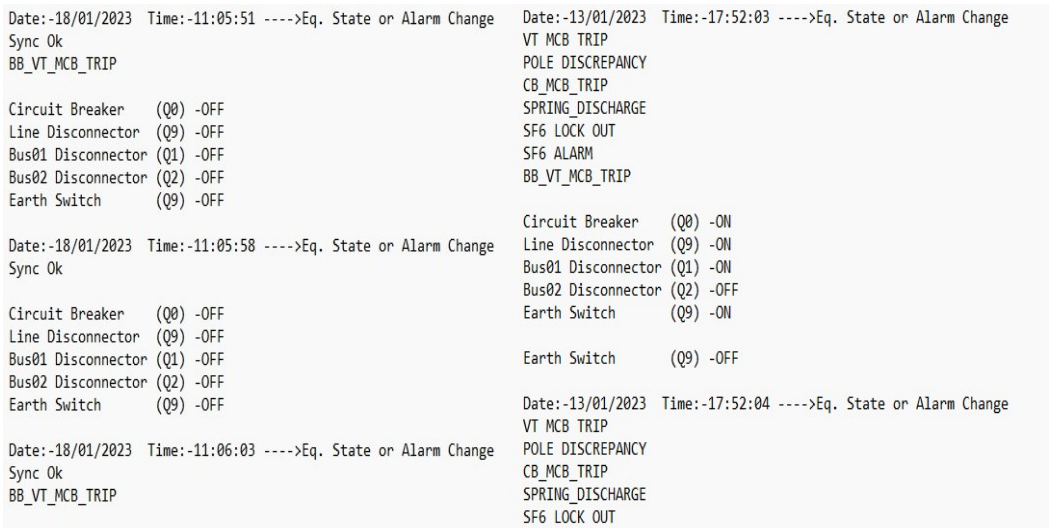


Figure 3.4 – Event Log Sample

4.1 Introduction to Network Connection

Network accessibility is also introduced for this unit for remote monitoring and remote data downloading functions. Arduino Ethernet shield was utilized for that function.

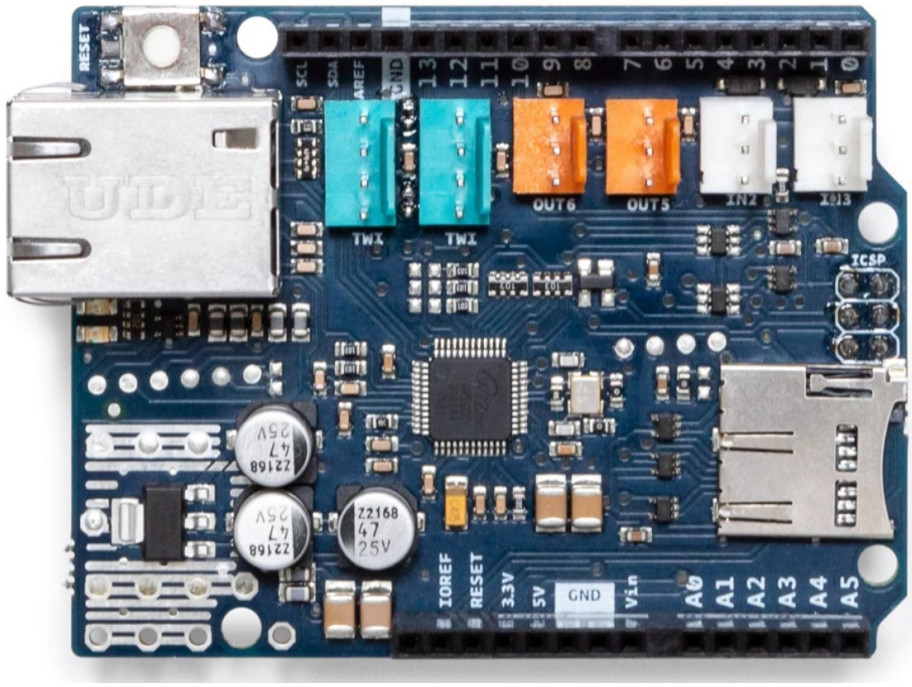


Figure 4.1 – Network Card

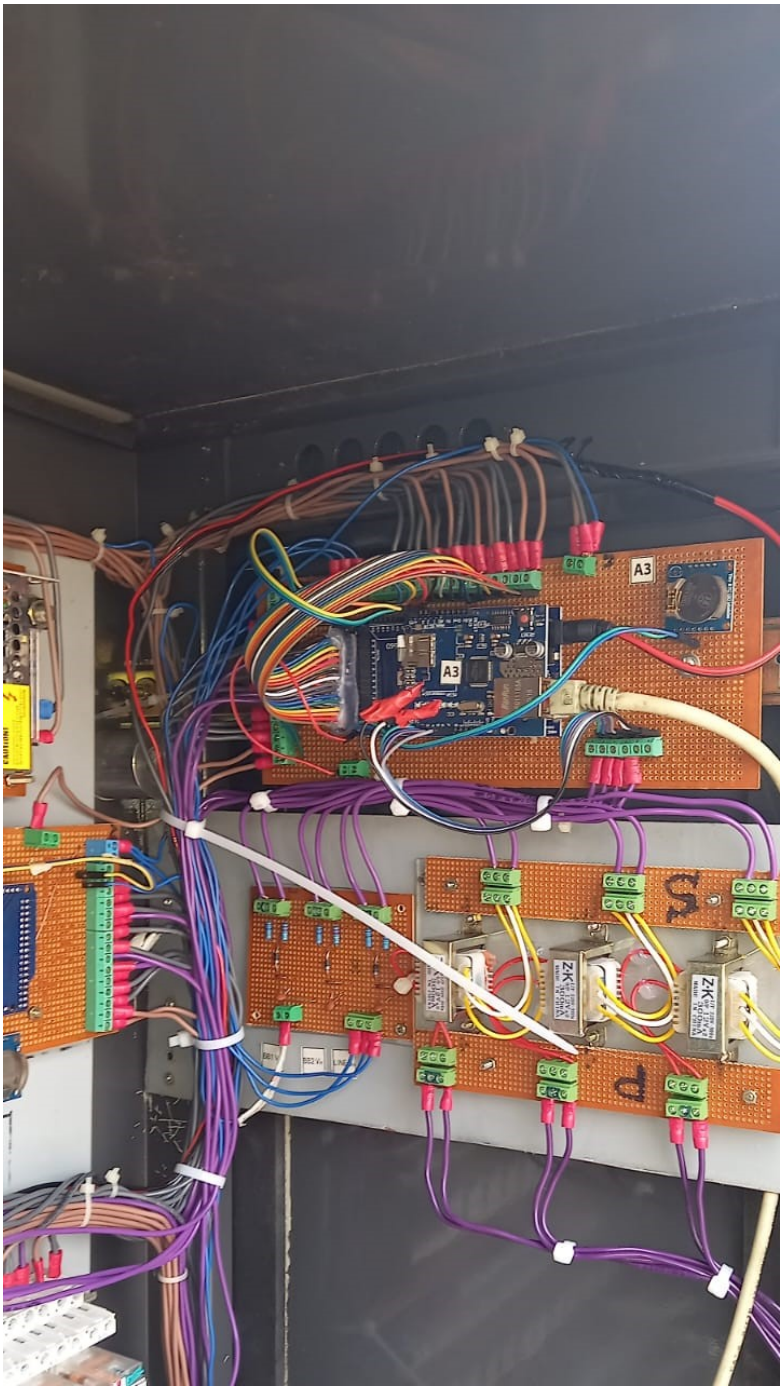


Figure 4.2– Network Card Position

4.2 User Interface

A network user interface was developed to access the unit. It was an HTML page with active Java components. These Java components are continuously updating equipment status. However, refreshing the web page is required to take the latest event record. The “Clear Event Memory” button is there to clear the memory after downloading the events to manage the SD card memory.

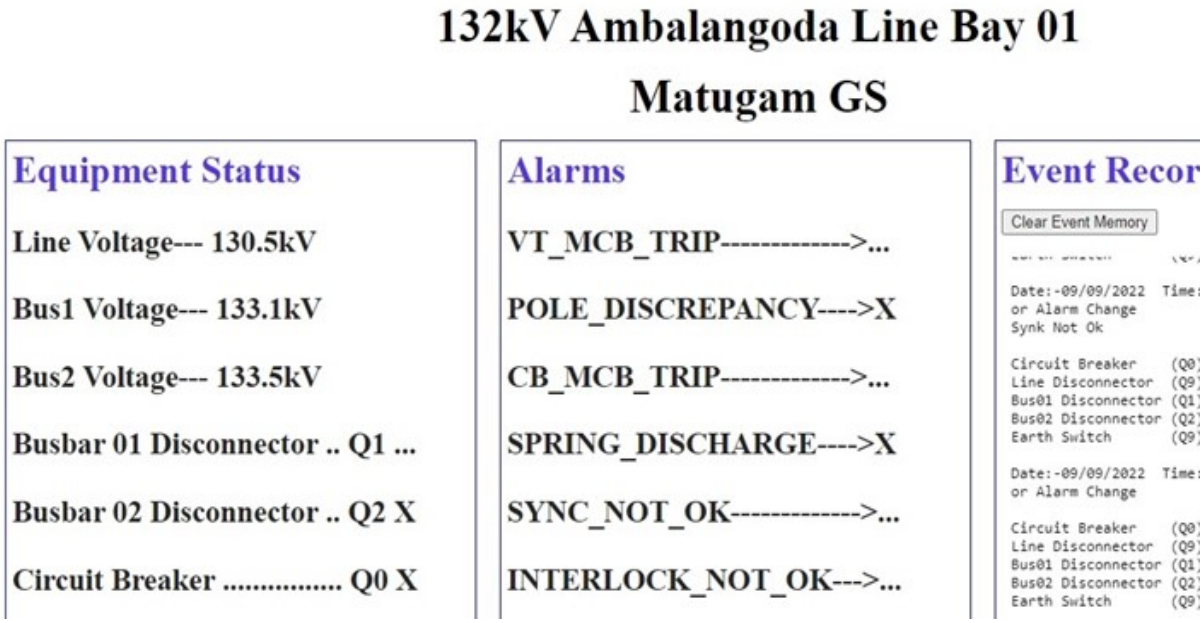


Figure 4.3 – User Interface

4.3 Network Connection Code Explanations

Network connection code is developed as per the flowchart given in figure 4.4.

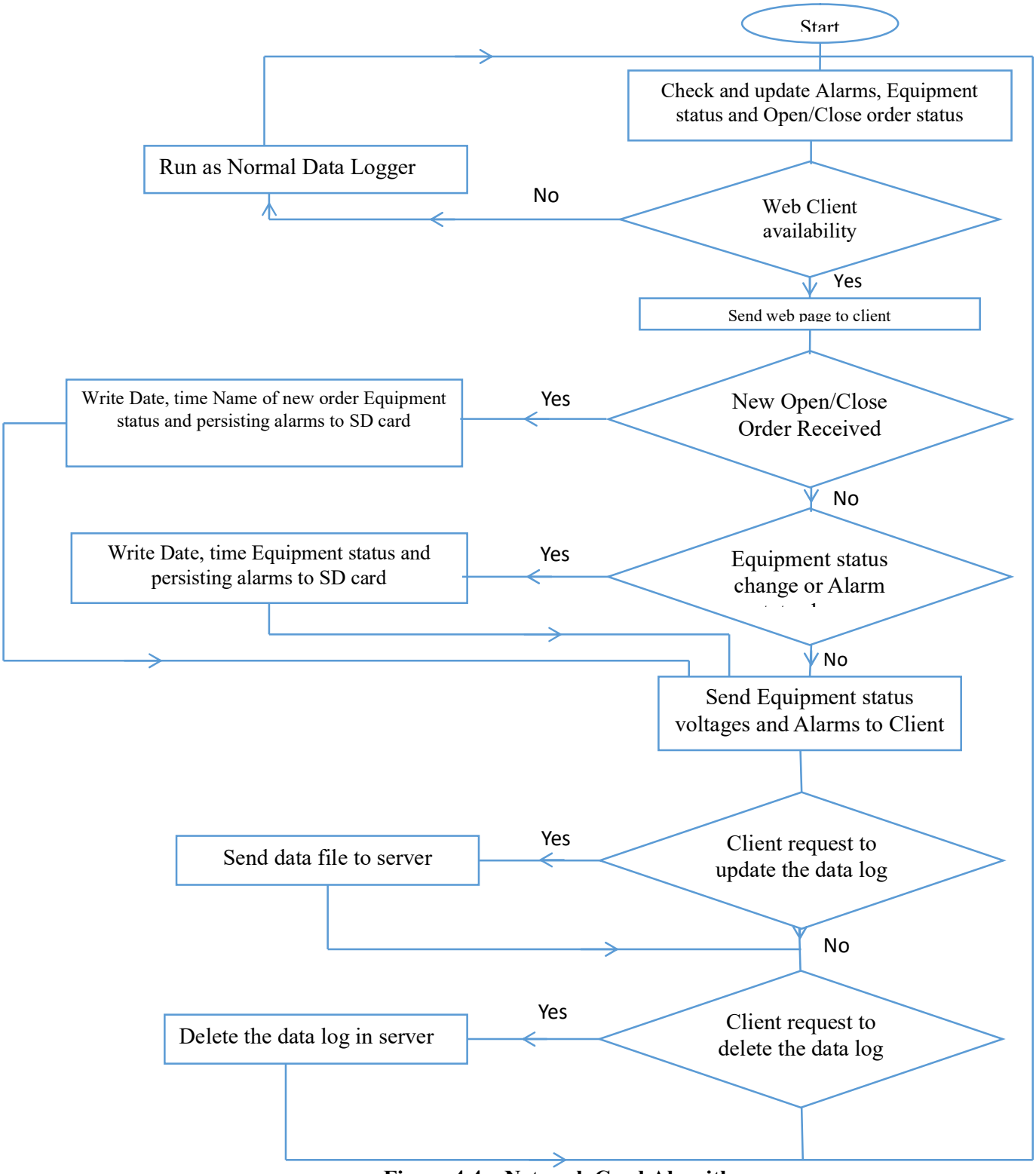


Figure 4.4 – Network Card Algorithm

5.1 Introduction to 132kV Circuit Breaker Operation Testing

Successful Circuit breaker close command required manual closing push button command, Switching Release, sync/energizing check release, and Close command timing.

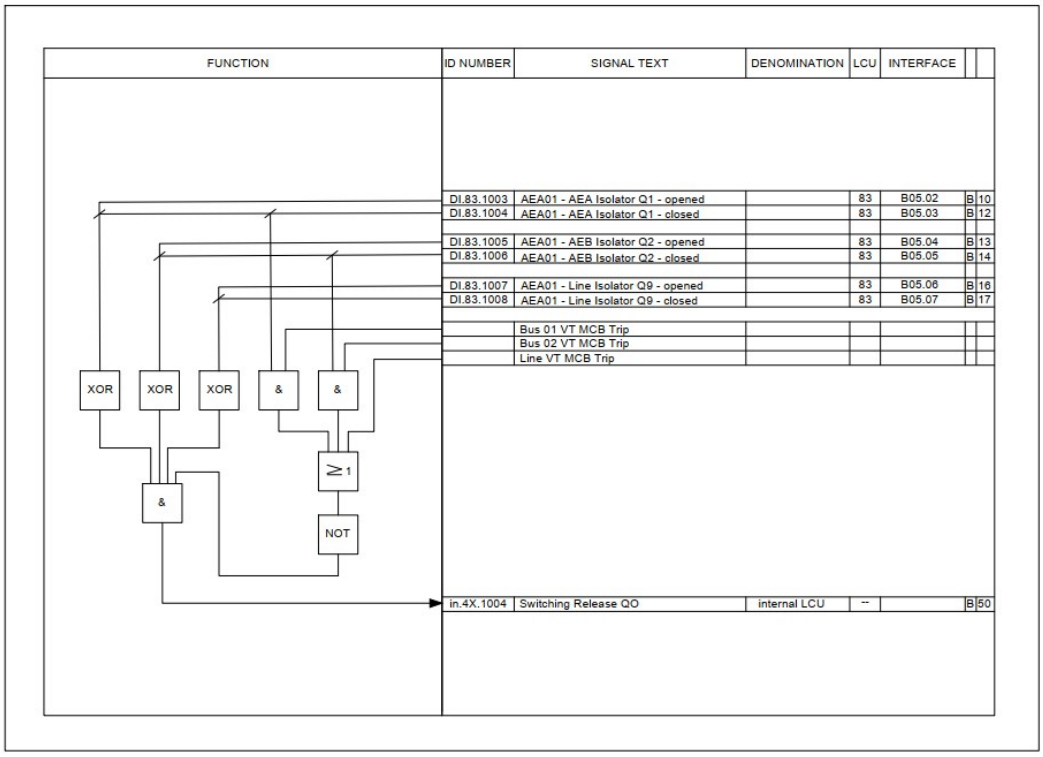
Separate outputs are configured for these Switching releases, sync/energizing check releases, and Close command timing. So, the statuses of those relays are visible to the outside and it will be very much helpful for testing processes.

Testing of the 132kV circuit breaker will continue by separating that into 3 parts as mentioned above to reduce the complexity of this most important equipment in the bay.

First, the Switching Release status will be checked and the sync/energizing check release will be checked later. Finally, complete Close command with Close command timing will be tested.

5.2 Switching Release Testing

Switching release testing will be done by operating Busbar 01 isolator, Busbar 02 isolator, Line isolator, and VT MCB statuses of Busbar 01, busbar 02 & line. The status of the Switching release relay be checked with the operation of above equipment as per the following table. The release signal shall be in accordance with the following logic diagrams provided by the C&P branch.



Q1 Opened	Q1 Closed	Q2 Opened	Q2 Closed	Q9 Opened	Q9 Closed	BB1VT Trip	BB2 VT Trip	LVT Trip	Expected Switch. Release	Actual Switch. Release
0	**0**	1	0	1	0	0	0	0	0	
0	**1**	1	0	1	0	0	0	0	1	
1	**0**	1	0	1	0	0	0	0	1	
1	**1**	1	0	1	0	0	0	0	0	
1	0	**0**	**0**	1	0	0	0	0	0	
1	0	**0**	**1**	1	0	0	0	0	1	
1	0	**1**	**0**	1	0	0	0	0	1	
1	0	**1**	**1**	1	0	0	0	0	0	
1	0	1	0	**0**	**0**	0	0	0	0	
1	0	1	0	**0**	**1**	0	0	0	1	
1	0	1	0	**1**	**0**	0	0	0	1	
1	0	1	0	**1**	**1**	0	0	0	0	
1	**0**	1	0	1	0	**0**	0	0	1	

1	**0**	1	0	1	0	**1**	0	0	1	
0	**1**	1	0	1	0	**0**	0	0	1	
0	**1**	1	0	1	0	**1**	0	0	0	
1	0	**1**	**0**	1	0	0	**0**	0	1	
1	0	**1**	**0**	1	0	0	**1**	0	1	
1	0	**0**	**1**	1	0	0	**0**	0	1	
1	0	**0**	**1**	1	0	0	**1**	0	0	
1	0	1	0	1	0	0	0	**1**	1	

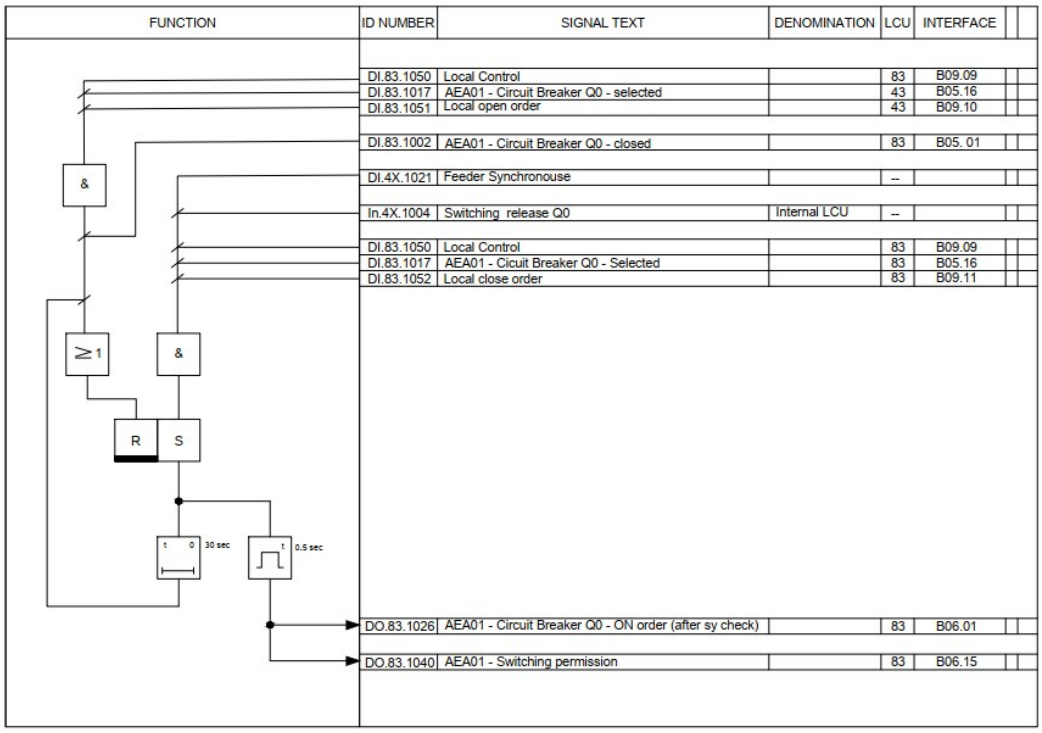
5.3 Synch/Energizing Release Check

The synch/energizing check release is provided by a separate synch check relay type KAVS100. The release is tested for the conditions specified in the following table.

Line V<0.2 pu	Line V>0.75 pu	BB1 V<0.2 pu	Line V<0.75 pu	Freq-diff < 0.1 Hz	Angle-diff < 20˚	Volt. Diff < 0.1 pu	Expected Synch Release	Actual Synch Release
0	0	**0**	0	0	0	0	0	
0	0	**1**	0	0	0	0	0	
1	0	**0**	0	0	0	0	0	
1	0	**1**	0	0	0	0	1	
0	**0**	0	**0**	0	0	0	0	
0	**0**	0	**1**	0	0	0	0	
0	**1**	0	**0**	0	0	0	0	
0	**1**	0	**1**	0	0	0	1	
0	1	0	1	**0**	**0**	**1**	0	
0	1	0	1	**0**	**1**	**0**	0	
0	1	0	1	**1**	**0**	**0**	0	

5.4 Close command with Close command time testing.

Close command with Close command time testing will be done by changing the Switching release, Synch/energizing check release and CB Closed state with manual CB opening command. The status of the CB Close relay in the MK box will be checked with the position changes of the above equipment as per the following table with manual Close order and Opening Orders. It shall be in accordance with the following logic diagram provided by the C&P branch.



Switching Release	Synch/ Energizing check	CB Closed Status	CB Opening Order	Inhibit time for repetitive close commands elapsed	Exp. CMD Status	Actual CMD Status
0	**0**	0	0	1	0	
0	**1**	0	0	1	0	
1	**0**	0	0	1	0	
1	**1**	0	0	1	1	
1	1	**1**	**0**	**1**	0	
1	1	**0**	**1**	**1**	0	
1	1	**0**	**0**	**0**	0	

5.5 Introduction to 132kV Busbar 1 and 2 Isolator Testing

Successful operation of Busbar Isolator close or open command required manual closing or opening push button command and Bus Isolator Switching Release.

Separate Switching Release Relays for Bus 01 isolator and Bus 2 isolator are configured in the bay unit. So, the statuses of those relays are visible to the outside and it will be very much helpful for testing processes.

5.6 Operation Testing of 132kV Busbar 1 Isolator

Successful operation of Busbar Isolator 01 close or open command required manual closing or opening command and Bus Isolator Switching Release.

Q0 opened Status	Q2 Isolator opened status	Bus coupler closed status	Q51 Bus Earth Switch opened status	Expected switch release	Actual switch release
1	0	0	1	0	
1	1	0	1	1	
0	1	0	1	0	
1	1	0	0	0	
0	0	1	1	1	
0	1	1	1	0	

5.7 Operation Testing of 132kV Busbar 2 Isolator

Successful operation of Busbar Isolator 02 close or open command required manual closing or opening push button command and Bus Isolator Switching Release.

Q0 opened Status	Q1 Isolator opened status	Bus coupler closed status	Q52 Bus Earth Switch opened status	Expected switch release	Actual switch release
1	0	0	1	0	
1	1	0	1	1	
0	1	0	1	0	
1	1	0	0	0	
0	0	1	1	1	
0	1	1	1	0	

5.8 Introduction to 132kV Line Isolator and Earth-switch Testing

Successful operation of Line Isolator close or open command required manual closing or opening push button command and Line Isolator Switching Release. Earth-switch operations are not allowed to be done remotely. The Bay unit only releases the crank inserting the solenoid as per the line Voltage.

5.9 Operation Testing of 132kV Line Isolator

Successful operation of Line Isolator close or open command required manual closing or opening push button command and Line Isolator Switching Release.

Q0 CB opened status	Q8 Earth switch opened status	Expected switch release	Actual switch release
0	0	0	
0	1	0	
1	0	0	
1	1	1	

5.10 Operation Testing of 132kV Earth-Switch

Earth-switch operations are not allowed to be done remotely. The Bay unit only releases the crank inserting the solenoid as per the line Voltage.

Q9 Line Isolator opened status	Line V. <20%	Expected switch release	Actual switch release
1	0	0	
1	1	1	
0	1	0	

Annexures

1. Drawings

110VDC

Local Remote SW.

Ardiuno Board

Close

Open

17/4:X03:9

17/4:X06:6

ON

RM1:R4

K24

K22

Q01 ON

RM1:R6

K22

K24

Q02 ON

RM1:R2

RM1:R9

K26

K27

Q0 ON ORDER

Q9 ON

RM1:R11

RM1:R15

K29

K30

Q8 ON
ES RELEASE

Q0 ON

RM1:R3

K23

K21

Q01 OFF

RM1:R5

K21

K23

Q02 OFF

K25

Q0 OFF ORDER

RM1:R10

K28

Q9 OFF

TITLE: BAYCONTROLLER LAYOUT		REV: 1.0
	Company: Tr.O&M-S, CEB	Sheet: 1/1
	Date: 2022-03-22 Drawn By:	

U7

ARDUINO MEGA

ARDUINO MEGA 2560

A02

A01

Power

Analog In

PWM

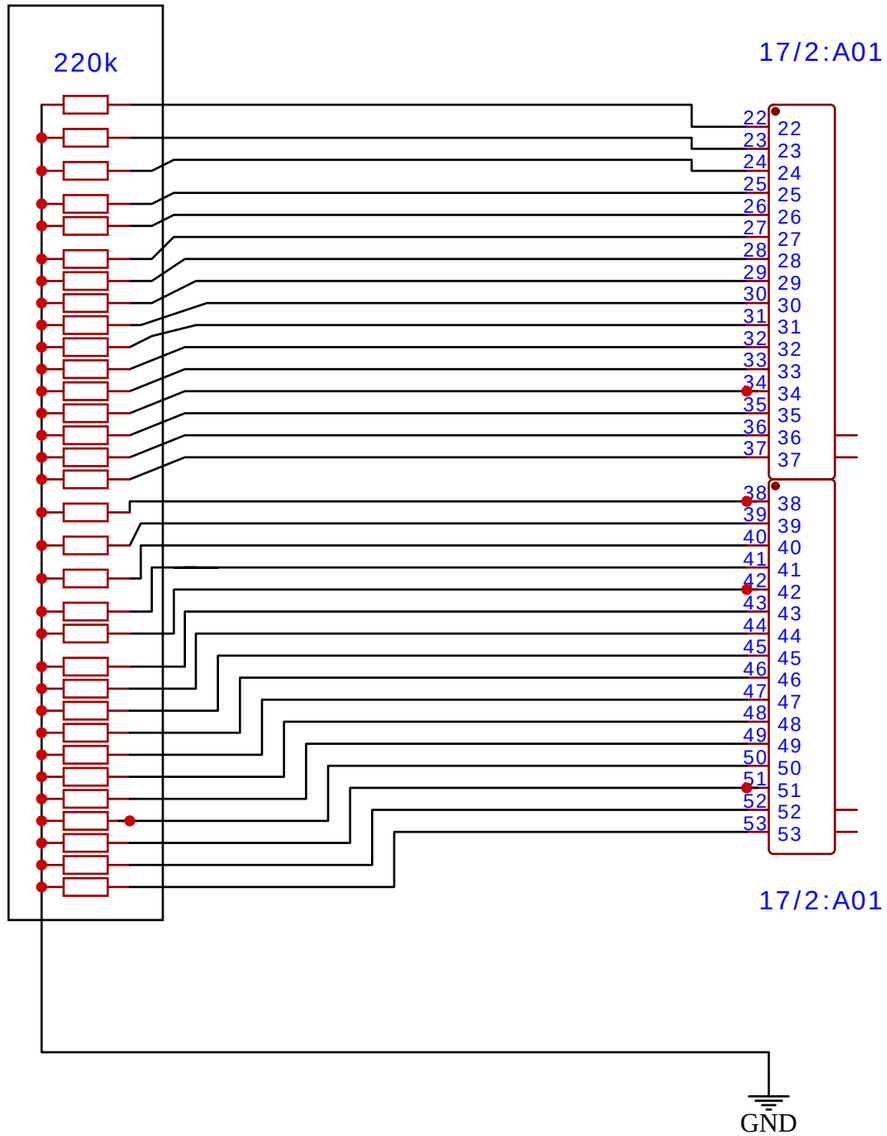
Communication

DIGITAL

17/4:RM-1

17/4:RM-2

TITLE:	ARDUINO BOARD
REV:	1.0
Company:	Tr.O&M-S, CEB
Sheet:	2/17
Date:	2022-03-22
Drawn By:	



TITLE: Arduino Inputs		REV: 1.0
	Company: Tr.O&M-S, CEB	Sheet: 3/17
	Date: 2022-03-29 Drawn By:	

+5V

GND

RM-2

In15 In14 In13 In12 In11 In10 In9 In8

15 14 13 12 11 10 9 8

RM-1

In7 In6 In5 In4 In3 In2 In1 In0

7 6 5 4 3 2 1 0

X06:6

17/11:H2:1

17/11:H5:1

X6

X3

X01

12

TITLE: RELAYMODULE CARD		REV: 1.0
	Company: Tr.O&M-S, CEB	Sheet: 4/17
	Date: 2022-03-22 Drawn By:	

A01:32
X2:6
A01:30
X2:5
A01:28
X2:4
A01:26
X2:3
A01:24
X2:2
A01:22
X2:1

K01 CB ON
K02 CB OFF
K03 B.DS1 ON
K04 B.DS1 OFF
K05 B.DS2 ON
K06 B.DS2 OFF

LED
Arduino +5Vdc

X04:01
X04:02
X04:03
X04:04
X04:05
X04:06

-110Vdc

TITLE: Input relays,110Vdc		REV: 1.0
	Company:Tr.O&M-S, CEB	Sheet: 5/17
	Date: 2022-03-29 Drawn By:	

A01:44

A01:42

A01:40

X2:10

A01:38

X2:9

A01:36

X2:8

A01:34

X2:7

K07 L.DS.ON

K08 L.DS.OFF

K09 ES.ON

K.10 ES.OFF

K.11 ES.BB1. ON

K.12 ES.BB1 OFF

LED

Arduino +5Vdc

X04:07

X04:08

X04:09

X04:10

X04:11

X04:12

-110Vdc

TITLE: Input relays,110Vdc		REV: 1.0
	Company: Tr.O&M-S, CEB	Sheet: 6/17
	Date: 2022-03-29 Drawn By:	

A01:51

A01:53

A01:52

A01:50

A01:48

A01:46

K.13
ES.BB2.ON

K.14
ES.BB2.OFF

K.15
SYN.CHK.
RELEASE ON

K.16
BUS.CUP.
ON

K.17
VT.MCB
TRIP

K.18
Spare

LED

Arduino +5Vdc

X04:13

X04:14

X04:15

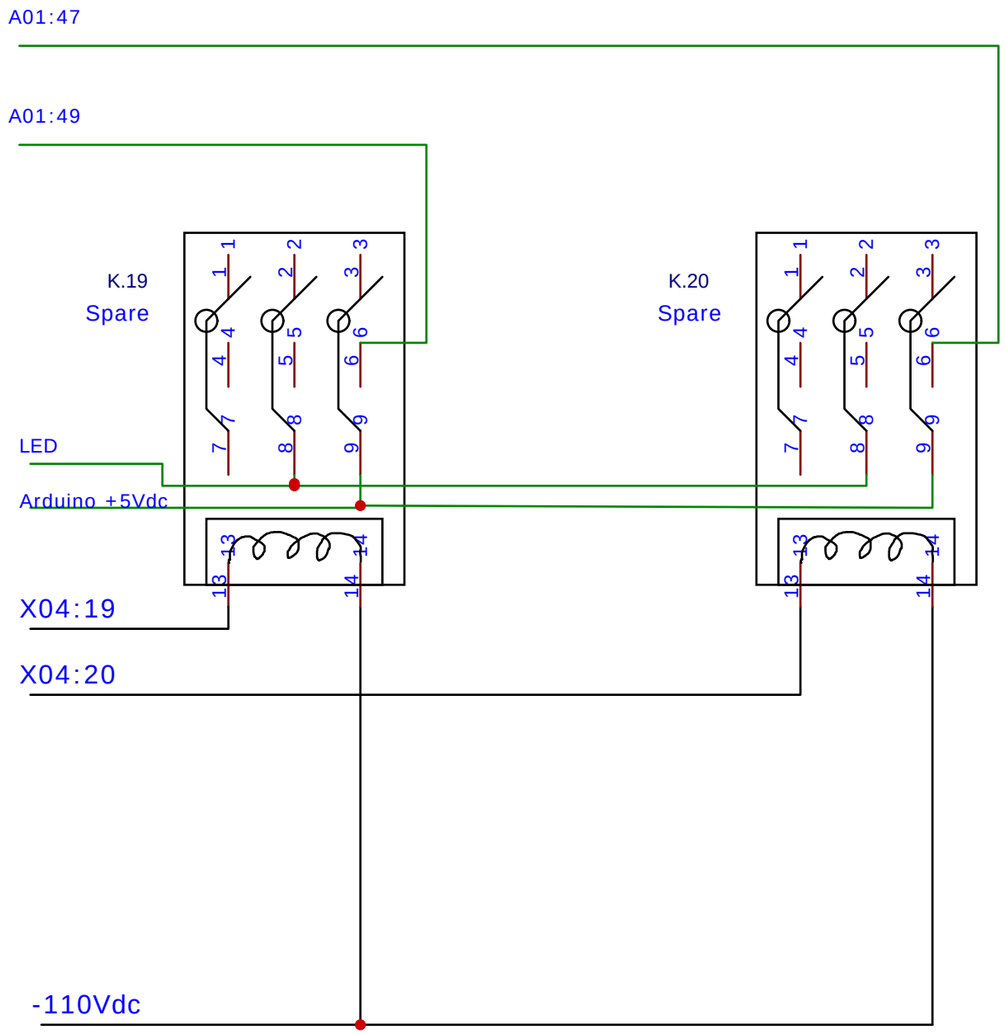
X04:16

X04:17

X04:18

-110Vdc

TITLE: Input relays,110Vdc		REV: 1.0
	Company: Tr.O&M-S, CEB	Sheet: 7/17
	Date: 2022-03-29 Drawn By:	



TITLE: Input relays,110Vdc		REV: 1.0
	Company: Tr.O&M-S, CEB	Sheet: 8/17
	Date: 2022-03-29 Drawn By:	

X7:12
X7:11
X7:10
X7:9
X7:8
X7:7
X7:6
X7:5
X7:4
X7:3
X7:1
X7:2

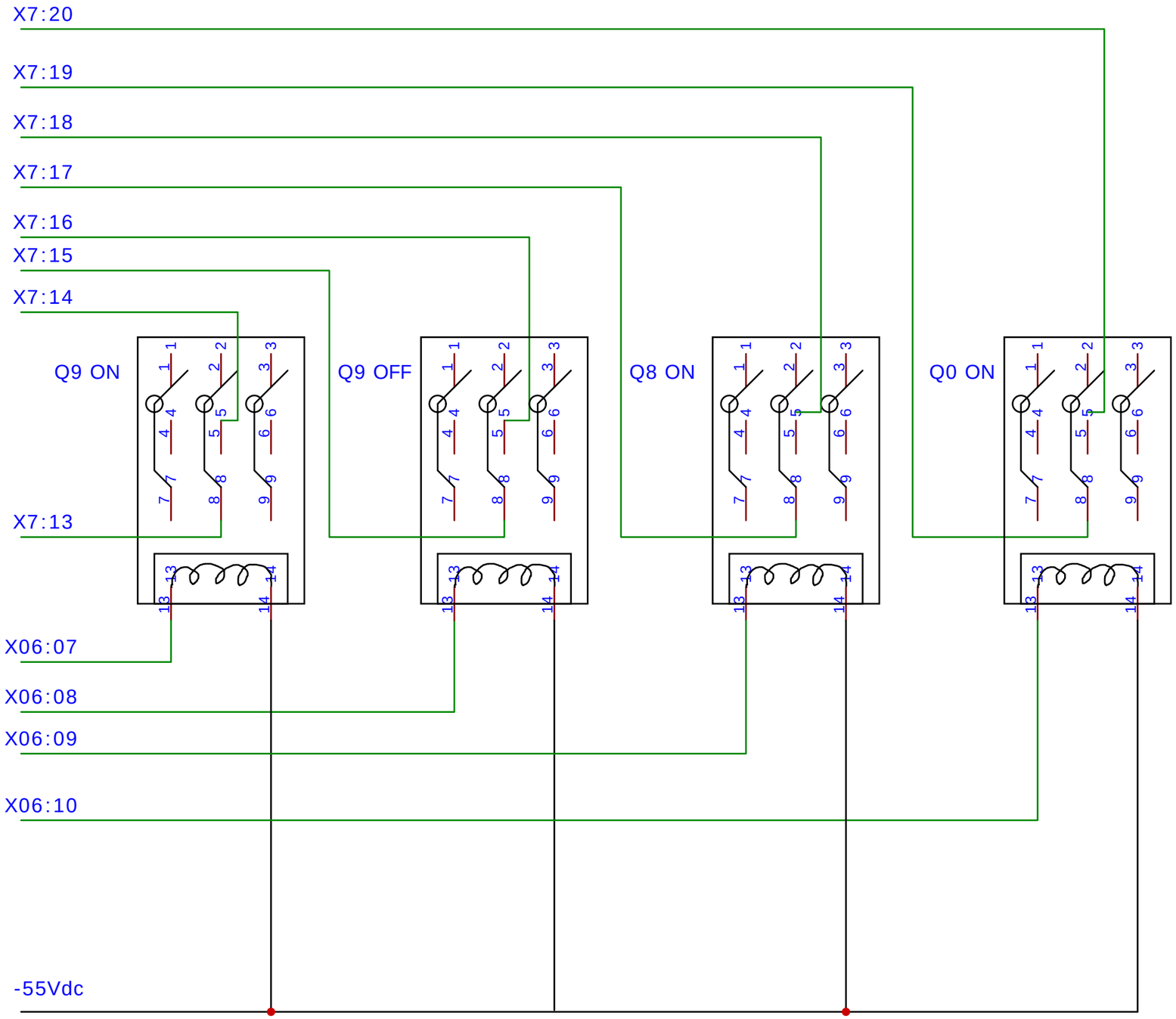
A0:45
A0:43

Q1 OFF
Q1 ON
Q2 OFF
Q2 ON
Q0 OFF ORDER
Q0 ON ORDER

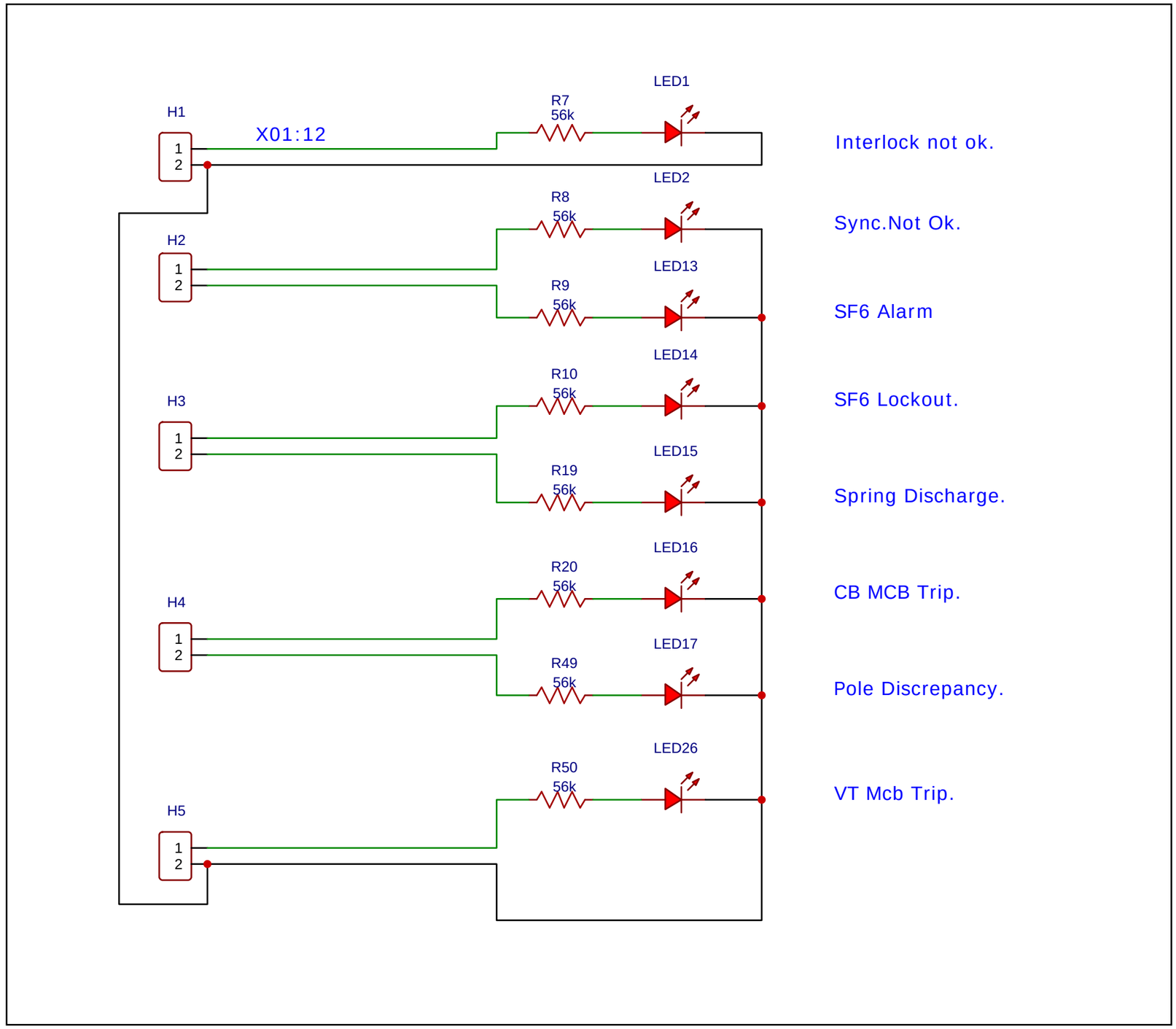
X06:01
X06:02
X06:03
X06:04
X06:05
X06:06

-55Vdc

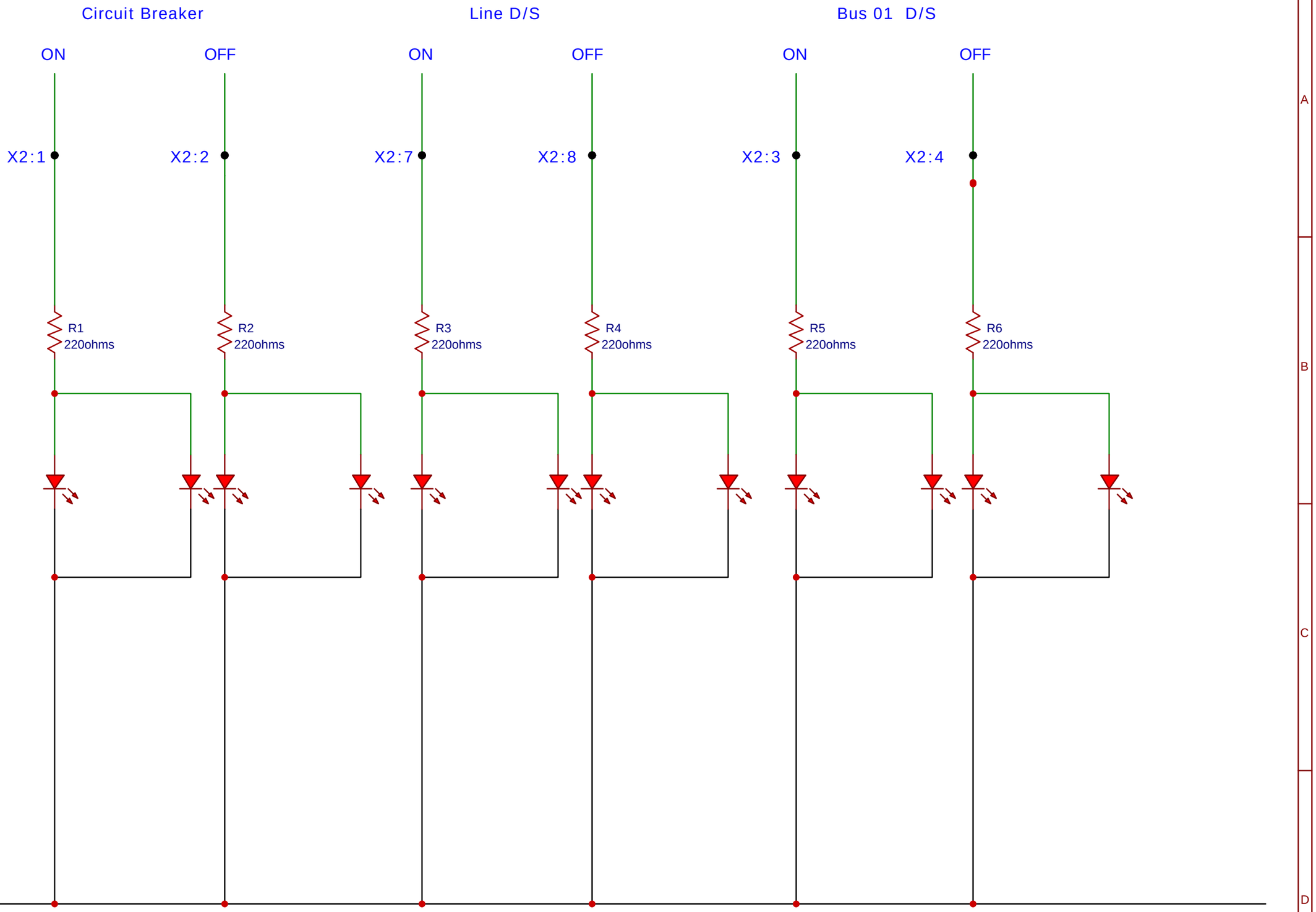
TITLE: Output relays,110Vdc		REV: 1.0
	Company: Tr.O&M-S, CEB	Sheet: 9/17
	Date: 2022-03-29 Drawn By:	



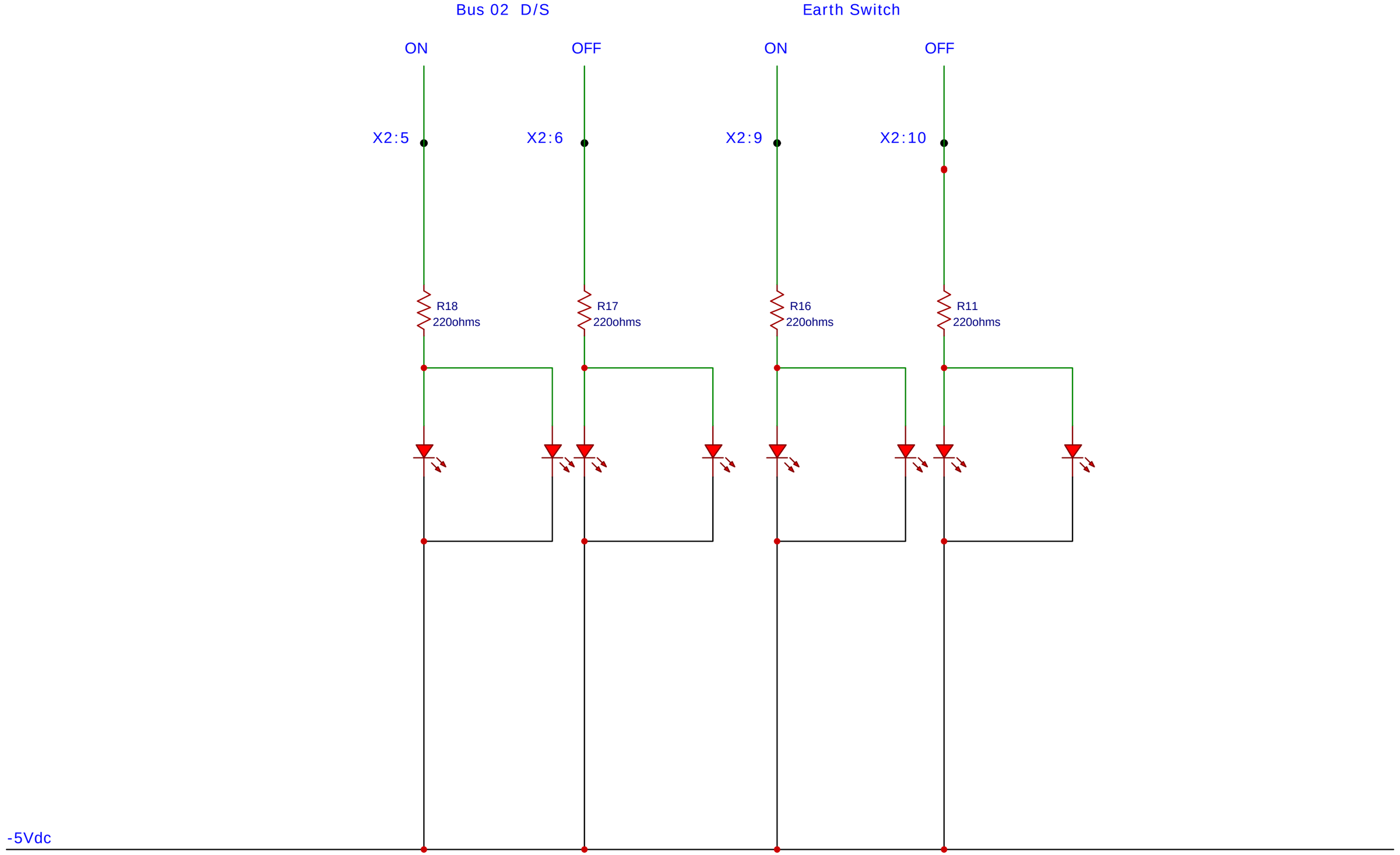
TITLE: Output relays,110Vdc		REV: 1.0
	Company: Tr.O&M-S, CEB	Sheet:10/17
	Date: 2022-04-08 Drawn By:	



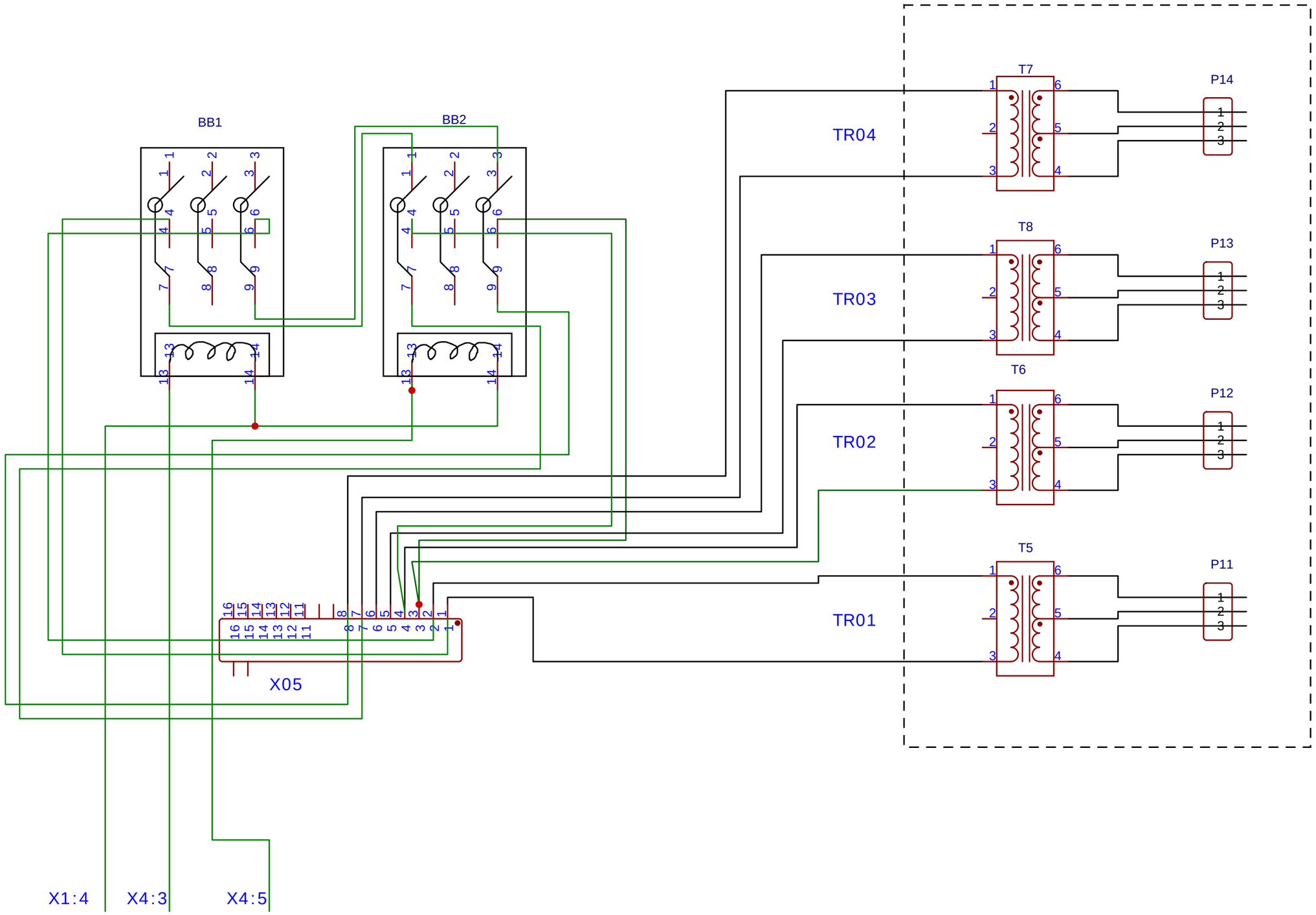
TITLE: 110VDC LED Indication		REV: 1.0
	Company: Tr.O&M-S, CEB	Sheet:11/17
	Date: 2022-03-29 Drawn By:	



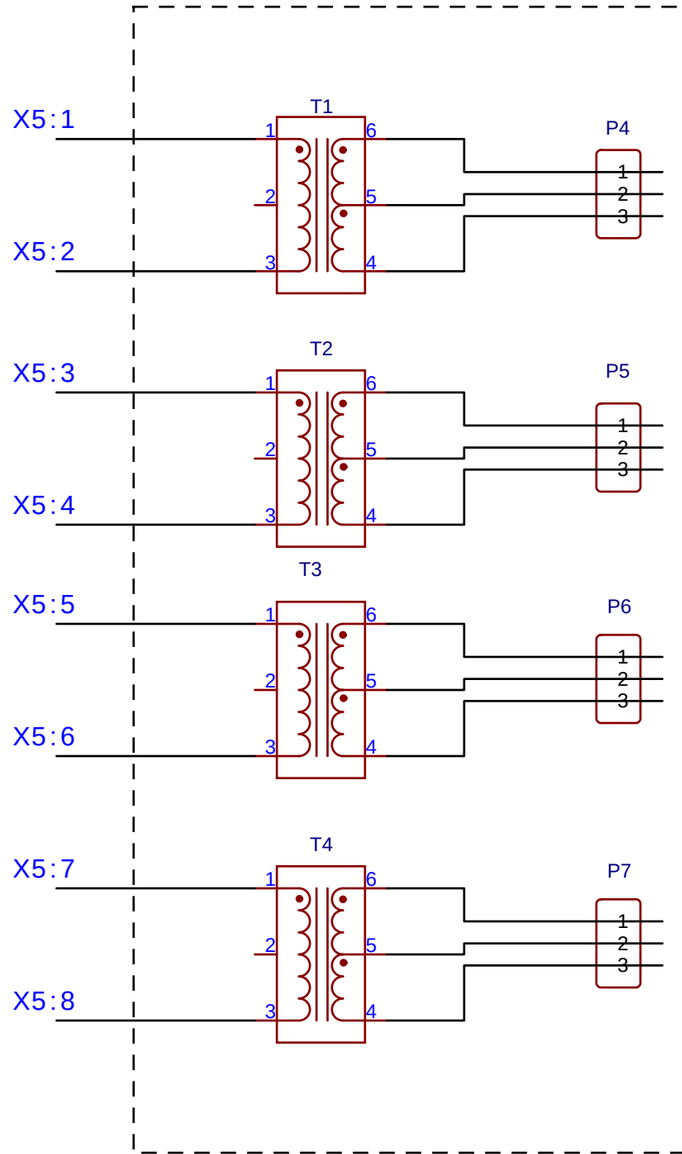
TITLE: Position Indications		REV: 1.0
	Company: Tr.O&M-S, CEB	Sheet:12/17
	Date: 2022-03-31 Drawn By:	



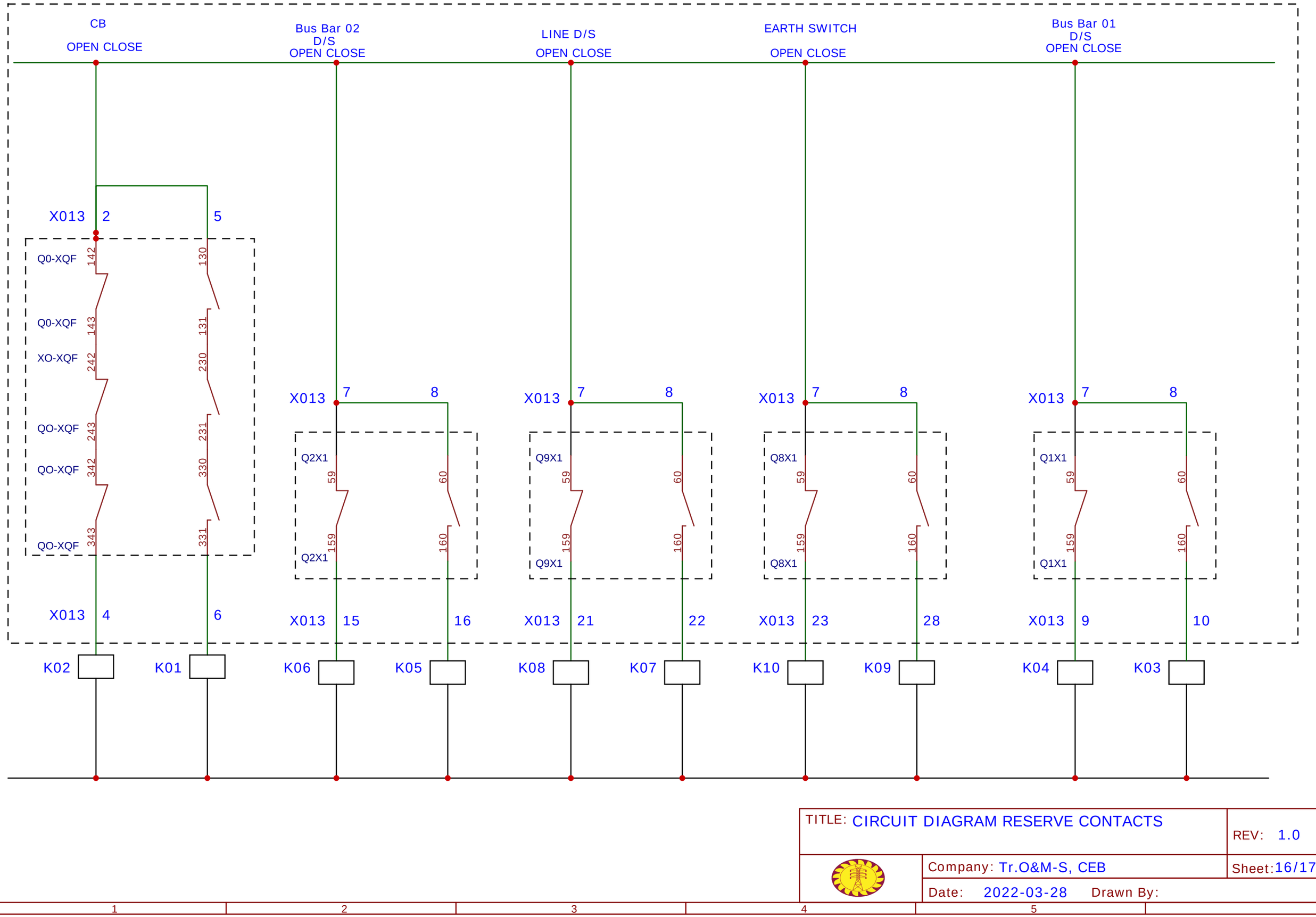
TITLE: Position Indications		REV: 1.0
	Company: Tr.O&M-S, CEB	Sheet:13/17
	Date: 2022-03-31 Drawn By:	



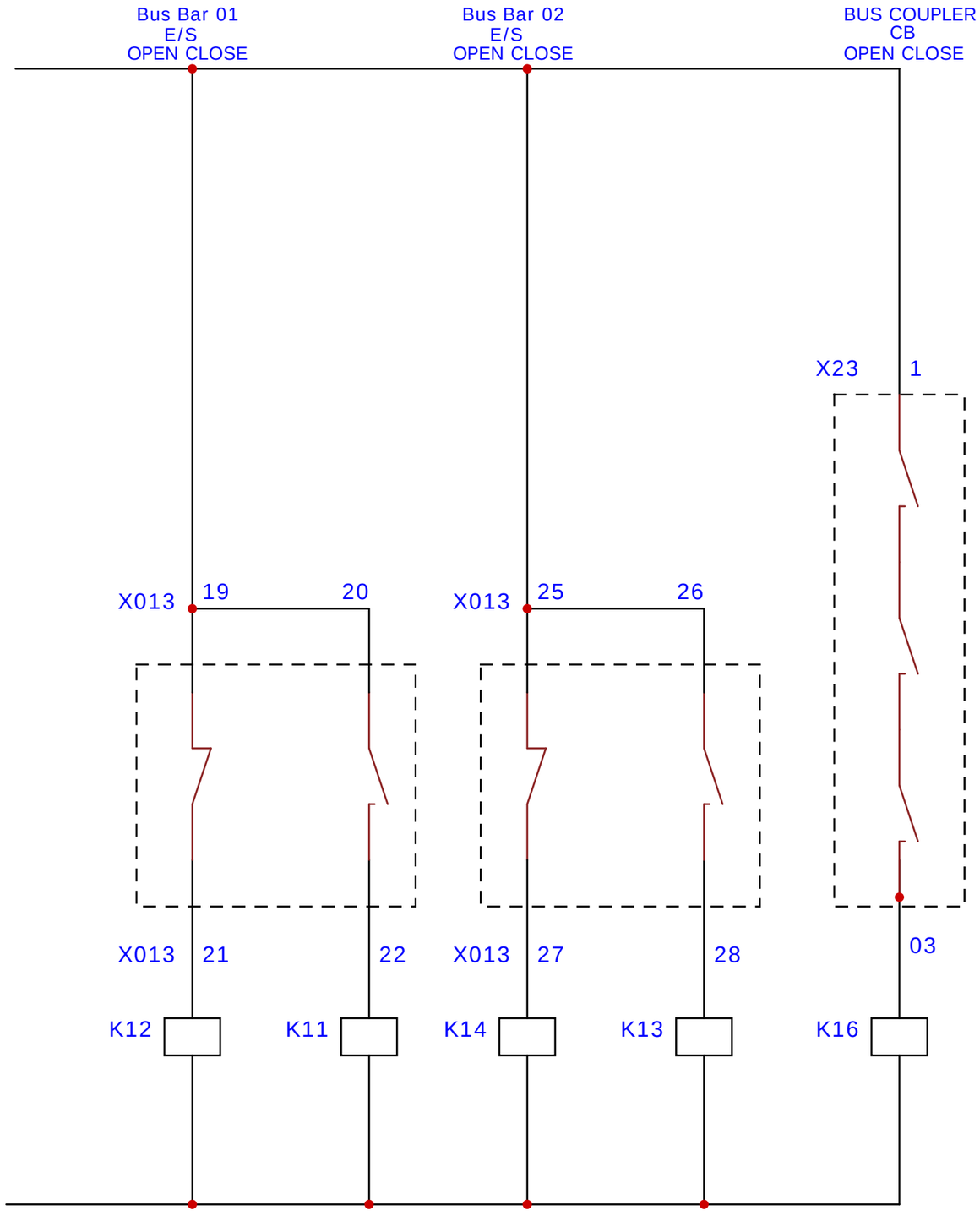
TITLE: Bus Bar Protection		REV: 1.0
	Company: Tr.O&M-S, CEB	Sheet:14/17
	Date: 2022-03-31 Drawn By:	



TITLE: Bus Bar Protection		REV: 1.0
	Company:Tr.O&M-S, CEB	Sheet:15/17
	Date: 2022-03-29 Drawn By:	



TITLE: CIRCUIT DIAGRAM RESERVE CONTACTS		REV: 1.0
	Company: Tr.O&M-S, CEB	Sheet:16/17
	Date: 2022-03-28 Drawn By:	



TITLE: CIRCUIT DIAGRAM RESERVE CONTACTS		REV: 1.0
	Company:Tr.O&M-S, CEB	Sheet:17/17
	Date: 2022-03-28 Drawn By:	

RGJayendra Add files via upload last year

525 lines (449 loc) · 22.5 KB

Code Blame Raw

```
1
2
3     #include "RTClib.h" // Date and time functions using a DS1307 RTC connected via I2C and Wire lib
4     #include <SPI.h>
5     #include <Ethernet.h>
6     #include <SD.h>
7     #define REQ_BUF_SZ   60 // size of buffer used to capture HTTP requests
8     char timedatebuf[65];  // Time and Date string buffer
9
10    ////////////////////////////////////////////////////////////////////////////////////////////////////////////////
11    // Unit Funtional Indicator
12      int Network_Controller        = 6;
13      int Network_Controller_RTC    = 7;
14      int Network_Controller_Memory = 8;
15      int Network_Connected         = 9;
16
17    ////////////////////////////////////////////////////////////////////////////////////////////////////////////////
18    // Operation orders
19      int CB_On_Order          = 22;
20      int CB_Off_Order         = 24;
21      int Line_Ds_On_Order     = 30;
22      int Line_Ds_Off_Order    = 32;
23      int B_Ds1_On_Order       = 38;
24      int B_Ds1_Off_Order      = 42;
25      int B_Ds2_On_Order       = 23;
26      int B_Ds2_Off_Order      = 25;
27     // int E_SW_Release         = 49;
28    // Equipment status input number  1B
29      int CB_On           = 26;
30      int CB_Off          = 28;
31      int Line_Ds_On      = 34;
32      int Line_Ds_Off     = 36;
33      int B_Ds_On         = 40;
34      int B_Ds_Off        = 44;
35      int B_Ds2_On        = 27;
36      int B_Ds2_Off       = 29;
37      int E_SW_On         = 46;
38      int E_SW_Off        = 48;
```

```
39      int Vt_Mcb_Trip     = 31;
40      int Pole_Discrepancy= 35;
41      int Cb_Mcb_Trip     = 37;
42      int Spring_Discharge= 39;
43      int Sf6_Lockout     = 41;
44      int Sf6_Alarm       = 43;
45      int Sync_Not_Ok     = 45;
46      int Interlock_Not_Ok = 47;
47
48    RTC_DS1307 rtc;
49
50    // Operation order variables   2A
51      boolean CB_ON_ORDER;
52      boolean CB_OFF_ORDER;
53      boolean LINE_DS_ON_ORDER;
54      boolean LINE_DS_OFF_ORDER;
55      boolean B_DS1_ON_ORDER;
56      boolean B_DS1_OFF_ORDER;
57      boolean B_DS2_ON_ORDER;
58      boolean B_DS2_OFF_ORDER;
59     // boolean E_SW_RELEASE;
60
61    // Equipment status variables   2B
62
63      boolean CB_ON;
64      boolean CB_OFF;
65      boolean LINE_DS_ON;
66      boolean LINE_DS_OFF;
67      boolean B_DS_ON;
68      boolean B_DS_OFF;
69      boolean B_DS2_ON;
70      boolean B_DS2_OFF;
71      boolean E_SW_ON;
72      boolean E_SW_OFF;
73      boolean VT_MCB_TRIP;
74      boolean POLE_DISCREPANCY;
75      boolean CB_MCB_TRIP;
76      boolean SPRING_DISCHARGE;
77      boolean SF6_LOCKOUT;
78      boolean SF6_ALARM;
79      boolean SYNC_NOT_OK;
80      boolean INTERLOCK_NOT_OK;
81
82    // Previous equipment status variables  3B
83
84      boolean CB_ON_P;
85      boolean CB_OFF_P;
86      boolean LINE_DS_ON_P;
87      boolean LINE_DS_OFF_P;
88      boolean B_DS_ON_P;
89      boolean B_DS_OFF_P;
90      boolean B_DS2_ON_P;
91      boolean B_DS2_OFF_P;
92      boolean E_SW_ON_P;
93      boolean E_SW_OFF_P;
```

```
94      boolean VT_MCB_TRIP_P;
95      boolean POLE_DISCREPANCY_P;
96      boolean CB_MCB_TRIP_P;
97      boolean SPRING_DISCHARGE_P;
98      boolean SF6_LOCKOUT_P;
99      boolean SF6_ALARM_P;
100     boolean SYNC_NOT_OK_P;
101     boolean INTERLOCK_NOT_OK_P;
102
103
104   // Status change variable
105
106     boolean STATE_CHANGE;
107
108
109
110   ////////////////////////////////////////////////////////////////////////////////////////////////////
111   // MAC address from Ethernet shield sticker under board
112     byte mac[] = { 0xA8, 0x61, 0x0A, 0xAE, 0x74, 0x8F };
113     IPAddress ip(169, 254, 237, 253); // IP address, may need to change depending on network
114     EthernetServer server(80);  // create a server at port 80
115     File webFile;               // the web page file on the SD card
116     char HTTP_req[REQ_BUF_SZ] = {0}; // buffered HTTP request stored as null terminated string
117     char req_index = 0;              // index into HTTP_req buffer
118
119
120   void setup(){
121
122       Serial.begin(9600);       // for debugging
123
124       pinMode(Network_Controller, OUTPUT);
125       pinMode(Network_Controller_RTC, OUTPUT);
126       pinMode(Network_Controller_Memory, OUTPUT);
127       pinMode(Network_Connected, OUTPUT);
128
129       digitalWrite(Network_Controller,HIGH);// Funtional Indicator
130
131       if (! rtc.begin()) {
132         Serial.println("Couldn't find RTC");
133         digitalWrite(Network_Controller_RTC,LOW);
134         Serial.flush();
135         while (1) delay(10);}
136         else {digitalWrite(Network_Controller_RTC,HIGH);}
137
138       if (! rtc.isrunning()) {
139         Serial.println("RTC is NOT running, let's set the time!");// When time needs to be set on a new devi
140         rtc.adjust(DateTime(F(__DATE__), F(__TIME__)));}  // following line sets the RTC to the date & time
141
142         // initialize SD card
143         Serial.println("Initializing SD card...");
144         if (!SD.begin(4)) {
145           Serial.println("ERROR - SD card initialization failed!");
146           digitalWrite(Network_Controller_Memory,LOW);
147           return; }   // init failed
148           else {digitalWrite(Network_Controller_Memory,HIGH);}
```

```
149
150           Serial.println("SUCCESS - SD card initialized.");
151           if (!SD.exists("index.htm")) {   // check for index.htm file
152             Serial.println("ERROR - Can't find index.htm file!");
153             return; } // can't find index file
154
155           Serial.println("SUCCESS - Found index.htm file.");
156
157           // Inputs
158           pinMode(40, INPUT);
159           pinMode(27, INPUT);
160           pinMode(26, INPUT);
161           pinMode(34, INPUT);
162           pinMode(46, INPUT);
163           pinMode(31, INPUT);
164           pinMode(35, INPUT);
165           pinMode(37, INPUT);
166           pinMode(39, INPUT);
167           pinMode(45, INPUT);
168           pinMode(47, INPUT);
169           pinMode(43, INPUT);
170           pinMode(41, INPUT);
171
172           Ethernet.begin(mac, ip);  // initialize Ethernet device
173           server.begin();           // start to listen for clients
174           Serial.print("server is at ");
175           Serial.println(Ethernet.localIP());}
176
177
178   void loop(){
179
180           EthernetClient client = server.available();  // try to get client
181
182           if (client) {  // got client?
183               boolean currentLineIsBlank = true;
184               while (client.connected()) {
185                   if (client.available()) {   // client data available to read
186                       char c = client.read(); // read 1 byte (character) from client
187                       digitalWrite(Network_Connected,HIGH);
188                       // limit the size of the stored received HTTP request
189                       // buffer first part of HTTP request in HTTP_req array (string)
190                       // leave last element in array as 0 to null terminate string (REQ_BUF_SZ - 1)
191                       if (req_index < (REQ_BUF_SZ - 1)) {
192                           HTTP_req[req_index] = c;          // save HTTP request character
193                           req_index++;
194                       }
195                       // last line of client request is blank and ends with \n
196                       // respond to client only after last line received
197                       if (c == '\n' && currentLineIsBlank) {
198                           // send a standard http response header
199                           client.println("HTTP/1.1 200 OK");
200                           // remainder of header follows below, depending on if
201                           // web page or XML page is requested
202                           // Ajax request - send XML file
```

```
                    if (StrContains(HTTP_req, "ajax_inputs")) {
                        // send rest of HTTP header
                        client.println("Content-Type: text/xml");
                        client.println("Connection: keep-alive");
                        client.println();
                        Delete_Events();
                        // send XML file containing input states
                        XML_response(client);


                    }else if (StrContains(HTTP_req, "GET /event.txt")) {
                        webFile = SD.open("event.txt");
                        if (webFile) {
                            client.println("HTTP/1.1 200 OK");
                            client.println();
                          while(webFile.available()) {
                                client.write(webFile.read()); // send web page to client
                            }
                            webFile.close();
                        }

                    } else {  // web page request
                        // send rest of HTTP header
                        client.println("Content-Type: text/html");
                        client.println("Connection: keep-alive");
                        client.println();
                        // send web page
                        webFile = SD.open("index.htm");        // open web page file
                        if (webFile) {
                            while(webFile.available()) {
                                client.write(webFile.read()); // send web page to client
                            }
                            webFile.close();
                        }
                    }
                    // display received HTTP request on serial port
                    Serial.print(HTTP_req);
                    // reset buffer index and all buffer elements to 0
                    req_index = 0;
                    StrClear(HTTP_req, REQ_BUF_SZ);
                    break;
                }
                // every line of text received from the client ends with \r\n
                if (c == '\n') {
                    // last character on line of received text
                    // starting new line with next character read
                    currentLineIsBlank = true;
                }
                else if (c != '\r') {
                    // a text character was received from client
                    currentLineIsBlank = false;
                }
            } // end if (client.available())
        } // end while (client.connected())
        delay(1);      // give the web browser time to receive the data
```

```
258              client.stop();} // close the connection
259              else {digitalWrite(Network_Connected,LOW);} // Funtional Indicator
260             // end if (client)
261
262     ////////////////////////////////////////////////////////////////////////////////////////////////////
263
264         STATE_CHANGE = LOW;
265
266     //Reading operation order and status input//////////////////////////////////
267
268         CB_ON_ORDER=digitalRead(CB_On_Order);
269         CB_OFF_ORDER=digitalRead(CB_Off_Order);///////////////4B
270         CB_ON=digitalRead(CB_On);
271         CB_OFF=digitalRead(CB_Off);//////////////////////////////////////5B
272         LINE_DS_ON_ORDER=digitalRead(Line_Ds_On_Order);
273         LINE_DS_OFF_ORDER=digitalRead(Line_Ds_Off_Order);
274         LINE_DS_ON=digitalRead(Line_Ds_On);
275         LINE_DS_OFF=digitalRead(Line_Ds_Off);
276         B_DS1_ON_ORDER=digitalRead(B_Ds1_On_Order);
277         B_DS1_OFF_ORDER=digitalRead(B_Ds1_Off_Order);
278         B_DS_ON=digitalRead(B_Ds_On);
279         B_DS_OFF=digitalRead(B_Ds_Off);
280         B_DS2_ON_ORDER=digitalRead(B_Ds2_On_Order);
281         B_DS2_OFF_ORDER=digitalRead(B_Ds2_Off_Order);
282         B_DS2_ON=digitalRead(B_Ds2_On);
283         B_DS2_OFF=digitalRead(B_Ds2_Off);
284     //  E_SW_RELEASE=digitalRead(E_SW_Release);
285         E_SW_ON=digitalRead(E_SW_On);
286         E_SW_OFF=digitalRead(E_SW_Off);
287         VT_MCB_TRIP=digitalRead(Vt_Mcb_Trip);
288         POLE_DISCREPANCY=digitalRead(Pole_Discrepancy);
289         CB_MCB_TRIP=digitalRead(Cb_Mcb_Trip);
290         SPRING_DISCHARGE=digitalRead(Spring_Discharge);
291         SF6_LOCKOUT=digitalRead(Sf6_Lockout);
292         SF6_ALARM=digitalRead (Sf6_Alarm);
293         SYNC_NOT_OK=digitalRead(Sync_Not_Ok);
294         INTERLOCK_NOT_OK=digitalRead(Interlock_Not_Ok);
295
296     //Detecting Equipment State change//////////////////////////////////////////////
297
298         if (not((CB_ON_P == CB_ON)and             //Q0
299            (CB_OFF_P == CB_OFF)))//////////////////6B
300            {STATE_CHANGE = HIGH;}
301
302         if (STATE_CHANGE == HIGH) {      //Updating state of equipment if state change true//////////////////
303             CB_ON_P = CB_ON;
304             CB_OFF_P = CB_OFF;}
305
306         if (not((LINE_DS_ON_P == LINE_DS_ON)and  //Q9
307               (LINE_DS_OFF_P == LINE_DS_OFF)))
308               {STATE_CHANGE =HIGH;}
309
310         if (STATE_CHANGE == HIGH){
311             LINE_DS_ON_P  = LINE_DS_ON;
312             LINE_DS_OFF_P = LINE_DS_OFF;}
```

```
312             LINE_DS_OFF_P = LINE_DS_OFF;}
313
314         if (not((B_DS_ON_P == B_DS_ON)and  //Q1
315               (B_DS_OFF_P == B_DS_OFF)))
316               {STATE_CHANGE =HIGH;}
317
318         if (STATE_CHANGE == HIGH){
319             B_DS_ON_P  = B_DS_ON;
320             B_DS_OFF_P = B_DS_OFF;}
321
322         if (not((B_DS2_ON_P == B_DS2_ON)and  //Q2
323               (B_DS2_OFF_P == B_DS2_OFF)))
324               {STATE_CHANGE =HIGH;}
325
326         if (STATE_CHANGE == HIGH){
327             B_DS2_ON_P  = B_DS2_ON;
328             B_DS2_OFF_P = B_DS2_OFF;}
329
330         if (not((E_SW_ON_P == E_SW_ON)and  //Q9
331               (E_SW_OFF_P == E_SW_OFF)))
332               {STATE_CHANGE =HIGH;}
333
334         if (STATE_CHANGE == HIGH){
335             E_SW_ON_P = E_SW_ON;
336             E_SW_OFF_P = E_SW_OFF;}
337
338         if (not((VT_MCB_TRIP_P == VT_MCB_TRIP)and  //Q2
339               (POLE_DISCREPANCY_P == POLE_DISCREPANCY)and
340               (CB_MCB_TRIP_P == CB_MCB_TRIP)and
341               (SPRING_DISCHARGE_P == SPRING_DISCHARGE)and
342               (SF6_LOCKOUT_P == SF6_LOCKOUT)and
343               (SF6_ALARM_P == SF6_ALARM)and
344               (SYNC_NOT_OK_P == SYNC_NOT_OK)and
345               (INTERLOCK_NOT_OK_P == INTERLOCK_NOT_OK)))
346               {STATE_CHANGE =HIGH;}
347
348         if (STATE_CHANGE == HIGH){
349             VT_MCB_TRIP_P  = VT_MCB_TRIP;
350             POLE_DISCREPANCY_P = POLE_DISCREPANCY;
351             CB_MCB_TRIP_P  = CB_MCB_TRIP;
352             SPRING_DISCHARGE_P  = SPRING_DISCHARGE;
353             SF6_LOCKOUT_P  = SF6_LOCKOUT;
354             SF6_ALARM_P  = SF6_ALARM;
355             SYNC_NOT_OK_P  = SYNC_NOT_OK;
356             INTERLOCK_NOT_OK_P = INTERLOCK_NOT_OK;}
357
358     //writng data to SD Card/////////////////////////////////////////////////////
359
360         if ((CB_ON_ORDER == HIGH)or (CB_OFF_ORDER == HIGH)or(STATE_CHANGE == HIGH)or
361            (LINE_DS_ON_ORDER == HIGH) or(LINE_DS_OFF_ORDER == HIGH) or
362            (B_DS1_ON_ORDER == HIGH) or(B_DS1_OFF_ORDER == HIGH) or
363            (B_DS2_ON_ORDER == HIGH) or(B_DS2_OFF_ORDER == HIGH)){
364
365            DateTime now = rtc.now();
366
```

```
367         //Circuit Breaker (Q0).4A
368            if (CB_ON_ORDER==HIGH)sprintf(timedatebuf,"Date:-%02d/%02d/%02d  Time:-%02d:%02d:%02d ---->Circuit
369            else if (CB_OFF_ORDER==HIGH)sprintf(timedatebuf,"Date:-%02d/%02d/%02d  Time:-%02d:%02d:%02d ---->Ci
370
371         //Line disconnector (Q9)
372            else if (LINE_DS_ON_ORDER == HIGH)sprintf(timedatebuf,"Date:-%02d/%02d/%02d Time:-%02d:%02d:%02d---
373            else if(LINE_DS_OFF_ORDER==HIGH)sprintf(timedatebuf,"Date:-%02d/%02d/%02d Time:-%02d:%02d:%02d---->
374
375         //bus disconnector(Q1)
376            else if (B_DS1_ON_ORDER == HIGH)sprintf(timedatebuf,"Date:-%02d/%02d/%02d Time:-%02d:%02d:%02d---->
377            else if(B_DS1_OFF_ORDER==HIGH)sprintf(timedatebuf,"Date:-%02d/%02d/%02d Time:-%02d:%02d:%02d---->Bu
378
379          //Earth Switch
380
381          //bus disconnector(Q2)
382            else if (B_DS2_ON_ORDER == HIGH)sprintf(timedatebuf,"Date:-%02d/%02d/%02d Time:-%02d:%02d:%02d---->
383            else if(B_DS2_OFF_ORDER==HIGH)sprintf(timedatebuf,"Date:-%02d/%02d/%02d Time:-%02d:%02d:%02d---->Bu
384            else if (STATE_CHANGE==HIGH)sprintf(timedatebuf,"Date:-%02d/%02d/%02d  Time:-%02d:%02d:%02d ---->Eq
385
386     ////////////////////////Creating file to write the event////////////////////////
387
388             File dataFile = SD.open("event.txt", FILE_WRITE);  // Open or Create file
389
390     ////////////////////////Writing Control Order to file//////////////////////////////
391
392           if (dataFile) {  // Check if file exist on SD Card
393                  dataFile.println(timedatebuf);
394                  Serial.println(timedatebuf);
395
396     ////////////////////////Equipment States/////////////////////////////////////////////////
397                if (VT_MCB_TRIP ==HIGH){                                    //vt mcb trip.
398                   dataFile.println ("VT MCB TRIP");
399                   Serial.println   ("VT MCB TRIP");}
400                if (POLE_DISCREPANCY==HIGH){                                //pole discrepancy.
401                   dataFile.println ("POLE DISCREPANCY");
402                   Serial.println   ("POLE DISCREPANCY");}
403                if (CB_MCB_TRIP==HIGH){                                    //cb mcb trip.
404                   dataFile.println ("CB_MCB_TRIP");
405                   Serial.println   ("CB_MCB_TRIP");}
406                if (SPRING_DISCHARGE==HIGH){                               //spring discharge.
407                   dataFile.println ("SPRING_DISCHARGE");
408                   Serial.println   ("SPRING_DISCHARGE");}
409                if (SF6_LOCKOUT==HIGH){                                    //sf.6 lockout
410                   dataFile.println ("SF6 LOCK OUT");
411                   Serial.println   ("SF6 LOCK OUT");}
412                if (SF6_ALARM==HIGH){                                      //sf6. alarm.
413                   dataFile.println ("SF6 ALARM");
414                   Serial.println   ("SF6 ALARM");}
415                if (SYNC_NOT_OK==HIGH){                                    //sync not ok.
416                   dataFile.println ("Sync Ok");
417                   Serial.println   ("Sync Ok");}
418                if (INTERLOCK_NOT_OK==HIGH){                               //interlock not ok.
419                   dataFile.println ("Inter Lock Not OK.");
420                   Serial.println   ("Inter Lock Not OK.");}
421                 if (CB ON == HIGH){                                        //Circuit Breaker(O0).
```

```
                  dataFile.println ("");
                  Serial.println ("");
                  dataFile.println ("Circuit Breaker    (Q0) -ON");
                  Serial.println   ("Circuit Breaker    (Q0) -ON");}
                if (CB_OFF == HIGH){
                  dataFile.println ("");
                  Serial.println ("");
                  dataFile.println ("Circuit Breaker    (Q0) -OFF");
                  Serial.println   ("Circuit Breaker    (Q0) -OFF");}
                if (LINE_DS_ON==HIGH){                                  //Line Disconnector(Q9).
                  dataFile.println ("Line Disconnector  (Q9) -ON");
                  Serial.println   ("Line Disconnector  (Q9) -ON");}
                if (LINE_DS_OFF==HIGH){
                  dataFile.println ("Line Disconnector  (Q9) -OFF");
                  Serial.println   ("Line Disconnector  (Q9) -OFF");}
                if (B_DS_ON==HIGH){                                     //Bus Disconnector(Q1).
                  dataFile.println ("Bus01 Disconnector (Q1) -ON");
                  Serial.println   ("Bus01 Disconnector (Q1) -ON");}
               if (B_DS_OFF==HIGH){
                  dataFile.println ("Bus01 Disconnector (Q1) -OFF");
                  Serial.println   ("Bus01 Disconnector (Q1) -OFF");}
                  if (B_DS2_ON==HIGH){                                  //Bus Disconnector(Q2).
                  dataFile.println ("Bus02 Disconnector  (Q2) -ON");
                  Serial.println   ("Bus02 Disconnector  (Q2) -ON");}
               if (B_DS2_OFF==HIGH){
                  dataFile.println ("Bus02 Disconnector (Q2) -OFF");
                  Serial.println   ("Bus02 Disconnector (Q2) -OFF");}
               if (E_SW_ON== HIGH){                                     //Earth Switch(Q9).
                  dataFile.println ("Earth Switch       (Q9) -ON");
                  Serial.println   ("Earth Switch       (Q9) -ON");
                  Serial.println ("");
                  dataFile.println ("");}
               if (E_SW_OFF== HIGH){
                   dataFile.println("Earth Switch       (Q9) -OFF");
                   Serial.println  ("Earth Switch       (Q9) -OFF");
                   Serial.println ("");
                   dataFile.println ("");}
    ////////////////////////Closing file after writing Equipment States/////////////////////

               dataFile.close();} else {   // Close file
                   Serial.println("error opening event.txt");}  }        // if file not on SD Card


        while ((digitalRead(CB_On_Order)== HIGH)or(digitalRead(CB_Off_Order)==HIGH)or     // wait until clear
            (digitalRead(Line_Ds_On_Order )== HIGH)or(digitalRead(Line_Ds_Off_Order )==HIGH)or
            (digitalRead(B_Ds1_On_Order )== HIGH)or(digitalRead(B_Ds1_Off_Order )==HIGH)or
            (digitalRead(B_Ds2_On_Order )== HIGH)or(digitalRead(B_Ds2_Off_Order )==HIGH))
            {}

     }//End of main loop

    ////////////////////////////////////////////////////////////////////////////////////////////////////
    // send the XML file with analog values, switch status
    void Delete_Events(void){
```

```
476         if (StrContains(HTTP_req,"Del_E")) {SD.remove("event.txt");}}
477
478     void XML_response(EthernetClient cl){
479         int analog_val;            // stores value read from analog inputs
480         int count;                 // used by 'for' loops
481         int sw_arr[] = {40, 27, 26, 34, 46, 31, 35, 37, 39, 45, 47, 43, 41};  // pins interfaced to switches
482
483         cl.print("<?xml version = \"1.0\" ?>");
484         cl.print("<inputs>");
485         // read analog inputs
486         for (count = 2; count <= 4; count++) { // A2 to A5
487             analog_val = analogRead(count);
488             cl.print("<analog>");
489             cl.print(analog_val);
490             cl.println("</analog>");}
491         // read switches
492         for (count = 0; count < 13; count++) {
493             cl.print("<switch>");
494             if (digitalRead(sw_arr[count])) {
495                 cl.print("X");}
496             else {
497                 cl.print("-");}
498             cl.println("</switch>");}
499         cl.print("</inputs>");}
500
501     // sets every element of str to 0 (clears array)
502     void StrClear(char *str, char length){
503         for (int i = 0; i < length; i++) {
504             str[i] = 0;}}
505
506     // searches for the string sfind in the string str
507     // returns 1 if string found
508     // returns 0 if string not found
509     char StrContains(char *str, char *sfind){
510         char found = 0;
511         char index = 0;
512         char len;
513
514         len = strlen(str);
515
516         if (strlen(sfind) > len) {
517             return 0;}
518         while (index < len) {
519             if (str[index] == sfind[found]) {
520                 found++;
521                 if (strlen(sfind) == found) {
522                     return 1;}}
523             else {found = 0;}
524             index++;}
525         return 0;}
```

132kVBayController / BayController16112022Ver04.ino

RGJayendra Add files via upload last year

370 lines (283 loc) · 12.5 KB

Code Blame Raw

```
1
2   //Time parameters for CB close*************************************************************************
3   int CB_CLS_TIME     = 50;
4   int CB_CLS_WAIT_TIME= 1000;  //Time delay until CB close
5   int CB_i            = 0;
6
7   //Voltage Setting value 20%****************************************************************************
8   int Vol_Set         =700;  // 20V test ok
9   unsigned int max_v  =0;
10
11  ////////////////////////////////////////////////////////////////////////////////////////////////////////
12  // Unit Funtional Indicator
13    int Logic_Controller       = 6;
14
15
16  //Defining inputs 22 to 53*****************************************************************************
17  const int CB_ON_IN              = 22;
18  const int CB_OFF_IN             = 24;
19  const int B_DS01_ON_IN          = 26;
20  const int B_DS01_OFF_IN         = 28;
21  const int B_DS02_ON_IN          = 30;
22  const int B_DS02_OFF_IN         = 32;
23  const int L_DS_ON_IN            = 34;
24  const int L_DS_OFF_IN           = 36;
25  const int ES_ON_IN              = 38;
26  const int ES_OFF_IN             = 40;
27  const int ES_BB1_ON_IN          = 42;
28  const int ES_BB1_OFF_IN         = 44;
29  const int ES_BB2_ON_IN          = 46;
30  const int ES_BB2_OFF_IN         = 48;
31  const int FEEDER_SYNC_IN        = 50;
32  const int BUS_COUP_ON_IN        = 52;
33  const int VT_MCBTRIP_ON_IN      = 53;
34  const int CB_OFF_ORDER_IN       = 45;  // defining by RGJ
35  const int CB_ON_ORDER_IN        = 41;
36  const int BB1_VT_MCBTRIP_ON_IN  = 39;
37  const int BB2_VT_MCBTRIP_ON_IN  = 37;
38
```

```
39
40
41  //Defining outputs 01 to 19 ***************************************************************************
42  const int CB_ON_CMD                     = 14;
43  const int CB_PRE_ON_CMD                 = 2;
44  const int BUS01_DSQ01_RELEASE_ON_CMD    = 3;
45  const int BUS01_DSQ01_RELEASE_OFF_CMD   = 4;
46  const int BUS02_DSQ02_RELEASE_ON_CMD    = 5;
47  const int BUS02_DSQ02_RELEASE_OFF_CMD   = 6;
48  const int L_DS_RELEASE_ON_CMD           = 19;
49  const int L_DS_RELEASE_OFF_CMD          = 20;
50  const int ES_RELEASE_ON_CMD             = 18;
51  const int SYNC_REQUIRED                 = 17;
52  const int VT_MCB_TRIP                   = 16;
53
54
55  //Defining veriables ***********************************************************************************
56  boolean CB_ON=0;
57  boolean CB_ON_ORDER=0;
58  boolean CB_OFF=0;
59  boolean CB_OFF_ORDER=0;
60  boolean B_DS01_ON=0;
61  boolean B_DS01_OFF=0;
62  boolean B_DS02_ON=0;
63  boolean B_DS02_OFF=0;
64  boolean L_DS_ON=0;
65  boolean L_DS_OFF=0;
66  boolean ES_ON=0;
67  boolean ES_OFF=0;
68  boolean ES_BB1_ON=0;
69  boolean ES_BB1_OFF=0;
70  boolean ES_BB2_ON=0;
71  boolean ES_BB2_OFF=0;
72  boolean SYN_CHK_RELEASE_ON=0;
73  boolean BUS_COUP_ON=0;
74  boolean LINE_VOL=0 ;
75  boolean BUS01_VOL=0;
76  boolean BUS02_VOL=0 ;
77  boolean VT_MCBTRIP_ON=0;
78  boolean BB1_MCBTRIP_ON=0;
79  boolean BB2_MCBTRIP_ON=0;
80  boolean FEEDER_SYNC=0;
81  boolean CB_SWITCHING_RELEASE=0;
82
83
84  //Defining veriables for CB ON cmd*********************************************************************
85  boolean CB_ON_A=0;
86  boolean CB_ON_B=0;
87  boolean CB_ON_C=0;
88  boolean CB_ON_D=0;
89  boolean CB_ON_E=0;
90  boolean CB_ON_F=0;
91  boolean CB_ON_G=0;
92  boolean CB_ON_H=0;
93  boolean CB_ON_I=0;
```

```
boolean CB_ON_J=0;
boolean CB_ON_K=0;
boolean CB_ON_L=0;
boolean CB_ON_M=0;
boolean CB_ON_N=0;
boolean CB_ON_O=0;
boolean CB_ON_P=0;
boolean CB_ON_Q=0;
boolean CB_ON_R=0;
boolean CB_ON_S=0;
boolean CB_ON_T=0;
boolean CB_ON_X=0;
boolean CB_ON_Y=0;
boolean CB_ON_Z=0;
boolean CB_ON_int=0;

//Defining veriables for analog inputs*********************************************************************

boolean Line_Voltage=0;
boolean BB1_Voltage=0;
boolean BB2_Voltage=0;
boolean Feeder_Voltage=0;

//Defining veriables for analog noise filter***************************************************************

#define BUF_SIZE 40

float array_calculate_avg(int * buf, int len);

int   buf[BUF_SIZE]   = {0};
int   buf_index       = 0;
float buf_avg         = 0.0;




void setup() {
  Serial.begin(9600);
  //Assigning inputs to digital IO*************************************************************************
  pinMode(CB_ON_IN, INPUT); // assigning inputs to digital IO
  pinMode(CB_OFF_IN, INPUT);
  pinMode(B_DS01_ON_IN, INPUT);
  pinMode(B_DS01_OFF_IN, INPUT);
  pinMode(B_DS02_ON_IN, INPUT);
  pinMode(B_DS02_OFF_IN, INPUT);
  pinMode(L_DS_ON_IN, INPUT);
  pinMode(L_DS_OFF_IN, INPUT);
  pinMode(ES_ON_IN, INPUT);
  pinMode(ES_OFF_IN, INPUT);
  pinMode(ES_BB1_ON_IN, INPUT);
  pinMode(ES_BB1_OFF_IN, INPUT);
  pinMode(ES_BB2_ON_IN, INPUT);
  pinMode(ES_BB2_OFF_IN, INPUT);
  pinMode(BUS_COUP_ON_IN, INPUT);
  pinMode(VT_MCBTRIP_ON_IN, INPUT);
```

```
149       pinMode(BB1_VT_MCBTRIP_ON_IN, INPUT);
150       pinMode(BB2_VT_MCBTRIP_ON_IN, INPUT);
151       pinMode(CB_ON_ORDER_IN, INPUT);
152       pinMode(CB_OFF_ORDER_IN, INPUT);
153       pinMode(FEEDER_SYNC_IN, INPUT);
154
155
156
157
158       //Assigning outputs to digital IO**************************************************************************
159       pinMode(CB_ON_CMD, OUTPUT);
160       pinMode(CB_PRE_ON_CMD, OUTPUT);
161       pinMode(BUS01_DSQ01_RELEASE_ON_CMD, OUTPUT);
162       pinMode(BUS01_DSQ01_RELEASE_OFF_CMD, OUTPUT);
163       pinMode(BUS02_DSQ02_RELEASE_ON_CMD, OUTPUT);
164       pinMode(BUS02_DSQ02_RELEASE_OFF_CMD, OUTPUT);
165       pinMode(L_DS_RELEASE_ON_CMD, OUTPUT);
166       pinMode(L_DS_RELEASE_OFF_CMD, OUTPUT);
167       pinMode(ES_RELEASE_ON_CMD, OUTPUT);
168       pinMode(SYNC_REQUIRED, OUTPUT);
169       pinMode(VT_MCB_TRIP, OUTPUT);
170
171
172      digitalWrite(CB_ON_CMD, HIGH);
173      digitalWrite(CB_PRE_ON_CMD, HIGH);
174      digitalWrite(BUS01_DSQ01_RELEASE_ON_CMD,HIGH);
175      digitalWrite(BUS01_DSQ01_RELEASE_OFF_CMD, HIGH);
176      digitalWrite(BUS02_DSQ02_RELEASE_ON_CMD, HIGH);
177      digitalWrite(BUS02_DSQ02_RELEASE_OFF_CMD, HIGH);
178      digitalWrite(L_DS_RELEASE_ON_CMD, HIGH);
179      digitalWrite(L_DS_RELEASE_OFF_CMD, HIGH);
180      digitalWrite(ES_RELEASE_ON_CMD, HIGH);
181      digitalWrite(SYNC_REQUIRED, HIGH);
182      digitalWrite(VT_MCB_TRIP, HIGH);
183
184
185       }
186
187   void loop() {
188
189     digitalWrite(Logic_Controller,HIGH);// Funtional Indicator
190     //Reading analog inputs ***********************************************************************************
191
192                 max_v=0;
193                 for(uint16_t i = 0; i <20; i++){
194                 uint16_t r =analogRead (A2);//Read From analog channel 2 (A2)
195                 if (max_v <r) max_v=r;
196                 delayMicroseconds (1);}
197                 if (BUF_SIZE == buf_index){buf_index = 0;}
198                 buf[buf_index++] = max_v;
199                 buf_avg = array_calculate_avg(buf, BUF_SIZE);
200
201
202     if (buf_avg<0) {Line_Voltage=LOW;}
```

```
203     else if (buf_avg<(Vol_Set)) {Line_Voltage=HIGH;}
204     else {Line_Voltage=LOW;}
205     Serial.println(buf_avg);
206
207
208
209     //Reading digital inputs ********************************************************************************
210     CB_ON = digitalRead(CB_ON_IN); // Reading digital inputs
211     CB_OFF = digitalRead(CB_OFF_IN);
212     B_DS01_ON=digitalRead(B_DS01_ON_IN);
213     B_DS01_OFF=digitalRead(B_DS01_OFF_IN);
214     B_DS02_ON=digitalRead(B_DS02_ON_IN);
215     B_DS02_OFF=digitalRead(B_DS02_OFF_IN);
216     L_DS_ON=digitalRead(L_DS_ON_IN);
217     L_DS_OFF=digitalRead(L_DS_OFF_IN);
218     ES_ON=digitalRead(ES_ON_IN);
219     ES_OFF=digitalRead(ES_OFF_IN);
220     ES_BB1_ON=digitalRead(ES_BB1_ON_IN);
221     ES_BB1_OFF=digitalRead(ES_BB1_OFF_IN);
222     ES_BB2_ON=digitalRead(ES_BB2_ON_IN);
223     ES_BB2_OFF=digitalRead(ES_BB2_OFF_IN);
224     BUS_COUP_ON=digitalRead(BUS_COUP_ON_IN);
225     VT_MCBTRIP_ON=digitalRead(VT_MCBTRIP_ON_IN);
226     BB1_MCBTRIP_ON=digitalRead(BB1_VT_MCBTRIP_ON_IN);
227     BB2_MCBTRIP_ON=digitalRead(BB2_VT_MCBTRIP_ON_IN);
228     CB_ON_ORDER=digitalRead(CB_ON_ORDER_IN);
229     CB_OFF_ORDER=digitalRead(CB_OFF_ORDER_IN);
230     FEEDER_SYNC=digitalRead(FEEDER_SYNC_IN);
231
232
233     //For testing by RGJ ************************************************************************************
234                                                         //Line_Voltage=HIGH;
235                                                         //BB1_Voltage=HIGH;
236                                                         //BB2_Voltage=HIGH;
237                                                         Feeder_Voltage=HIGH;
238
239
240
241
242     //Q1 Open/Close Release Operation*************************************************************************
243
244     if  (((B_DS02_ON) && not(B_DS02_OFF) && not(ES_BB1_ON) && (ES_BB1_OFF) && (BUS_COUP_ON))
245     or ( (CB_OFF) && not(CB_ON)&&(B_DS02_OFF) && not (B_DS02_ON ) && not (ES_BB1_ON) && (ES_BB1_OFF))){  //(CB
246
247       digitalWrite(BUS01_DSQ01_RELEASE_ON_CMD,LOW);
248       digitalWrite(BUS01_DSQ01_RELEASE_OFF_CMD,LOW);
249     }
250     else{digitalWrite(BUS01_DSQ01_RELEASE_ON_CMD,HIGH);
251          digitalWrite(BUS01_DSQ01_RELEASE_OFF_CMD,HIGH);}
252
253
254     //Q2 Open/Close Release Operation*************************************************************************
255
256     if  (((B_DS01_ON) && not(B_DS01_OFF) && not(ES_BB2_ON) && (ES_BB2_OFF) && (BUS_COUP_ON))
257     or ((CB_OFF) && not(CB_ON) && (B_DS01_OFF) && not (B_DS01_ON ) && not (ES_BB2_ON) && (ES_BB2_OFF))){
```

```
258
259       digitalWrite(BUS02_DSQ02_RELEASE_ON_CMD,LOW);
260       digitalWrite(BUS02_DSQ02_RELEASE_OFF_CMD,LOW);
261     }
262     else{digitalWrite(BUS02_DSQ02_RELEASE_ON_CMD,HIGH);
263          digitalWrite(BUS02_DSQ02_RELEASE_OFF_CMD,HIGH);}
264
265
266     //Q8 - ES Release Operation*************************************************************************
267
268     if  ( Line_Voltage && L_DS_OFF && not(L_DS_ON) && not (VT_MCBTRIP_ON))
269           {digitalWrite(ES_RELEASE_ON_CMD,LOW);}
270     else  {digitalWrite(ES_RELEASE_ON_CMD,HIGH);}
271
272
273     //Q0 Close ***************************************************************************************
274
275     // Switching release Q0 as per the logic given by PD-CEB
276
277     CB_ON_A = (B_DS01_OFF xor B_DS01_ON);
278
279     CB_ON_B = (B_DS02_OFF xor B_DS02_ON);
280
281     CB_ON_C = (L_DS_OFF xor L_DS_ON);
282
283     CB_ON_E = (B_DS01_ON && BB1_MCBTRIP_ON) ;
284
285     CB_ON_F = (B_DS02_ON && BB2_MCBTRIP_ON) ;
286
287     CB_ON_G = (VT_MCBTRIP_ON) ;
288
289     CB_ON_D = (CB_ON_E or CB_ON_F or CB_ON_G) ;
290
291     CB_SWITCHING_RELEASE = (CB_ON_A && CB_ON_B && CB_ON_C && not(CB_ON_D)) ;
292
293     if (CB_SWITCHING_RELEASE) { digitalWrite(CB_PRE_ON_CMD,LOW); } else { digitalWrite(CB_PRE_ON_CMD,HIGH);}
294
295
296               /// Final CB_ON_CMD
297                 if ((CB_ON_ORDER or CB_ON_int) && (FEEDER_SYNC) && CB_SWITCHING_RELEASE &&
298                          (not(CB_OFF_ORDER))&&(not(CB_ON)) ) {
299                         CB_ON_int=1;
300                         CB_i++;
301
302                         if (CB_i<CB_CLS_TIME){digitalWrite(CB_ON_CMD,LOW);}
303                         else { digitalWrite(CB_ON_CMD,HIGH);}
304
305                         if (CB_CLS_WAIT_TIME<CB_i){CB_ON_int=0;}
306
307                 }else if (CB_OFF_ORDER or CB_ON) {
308                  digitalWrite(CB_ON_CMD,HIGH);
309                  CB_ON_int=0;
310                  CB_i=0;}
311
312                  else {
```

```
312                 else {
313                   digitalWrite(CB_ON_CMD,HIGH);
314                   CB_ON_int=0;
315                   CB_i=0;}
316
317     //        //Testing of CB ON Order
318     //
319     //        if (CB_ON_ORDER) {
320     //          Serial.println("CB_ON_ORDER");
321     //          if (CB_SWITCHING_RELEASE){Serial.println("CB_SWITCHING_RELEASE");}
322     //          if (not(CB_OFF_ORDER)){Serial.println("No CB Off Order");}
323     //          if (not(CB_ON)){Serial.println("No CB On");}
324     //          else {Serial.println("CB On");}}
325     //
326     //
327     //        if (FEEDER_SYNC){Serial.println("Feeder Sync ok");}
328     //
329     //         delayMicroseconds (500);
330
331
332
333
334     //Q9 Release (L  D/S)*************************************************************************************
335
336     if ((CB_OFF) && not(CB_ON) && not(ES_ON) && (ES_OFF)){
337        digitalWrite(L_DS_RELEASE_ON_CMD, LOW);
338        digitalWrite(L_DS_RELEASE_OFF_CMD, LOW);}
339        else {
340       digitalWrite(L_DS_RELEASE_ON_CMD, HIGH);
341       digitalWrite(L_DS_RELEASE_OFF_CMD,HIGH);}
342
343      //Sync ok signal*****************************************************************************************
344     if (FEEDER_SYNC){
345        digitalWrite(SYNC_REQUIRED, LOW);}
346        else {
347        digitalWrite(SYNC_REQUIRED,HIGH);}
348
349     //VT MCB on signal***************************************************************************************
350
351     if (VT_MCBTRIP_ON){
352        digitalWrite(VT_MCB_TRIP, LOW);}
353        else {
354        digitalWrite(VT_MCB_TRIP,HIGH);}
355
356
357       }
358
359     float array_calculate_avg(int * buf, int len)
360     {
361         int sum = 0;
362
363         for (int j=0; j<len; j++) {
364           sum+=buf[j];
365         }
366
```

```
367       return ((float)sum)/((float)len);
368      }
369
370
```

132kVBayController / index.htm

RGJayendra Add files via upload last year

117 lines (106 loc) · 3.62 KB

Code Blame Raw

```
<!DOCTYPE html>
<!-- saved from url=(0064)file:///C:/Users/Admin/Downloads/web_server_IO%20-%202/index.htm -->
<html><head><meta http-equiv="Content-Type" content="text/html; charset=windows-1252">
        <title>Amb01_Bay_Unit</title>
        <script>
                strDEL_E = "";
                function GetArduinoIO()
                {
                        nocache = "&nocache=" + Math.random() * 1000000;
                        var request = new XMLHttpRequest();
                        request.onreadystatechange = function()
                        {
                                if (this.readyState == 4) {
                                        if (this.status == 200) {
                                                if (this.responseXML != null) {
                                                        // XML file received - contains analog values, swi
                                                        var count;
                                                        // get analog inputs
                                                        var num_an = this.responseXML.getElementsByTagName
                                                        for (count = 0; count < num_an; count++) {
                                                                document.getElementsByClassName("analog")[
                                                                        this.responseXML.getElementsByTagN
                                                        }
                                                        // get switch inputs
                                                        var num_an = this.responseXML.getElementsByTagName
                                                        for (count = 0; count < num_an; count++) {
                                                                document.getElementsByClassName("switches"
                                                                        this.responseXML.getElementsByTagN
                                                        }
                                                }
                                        }
                                }
                        }
                        // send HTTP GET request with LEDs to switch on/off if any
                        request.open("GET", "ajax_inputs" + strDEL_E +  nocache, true);
                        request.send(null);
                        setTimeout('GetArduinoIO()', 1000);
                        strDEL_E = "";
```

```
                }

                function GetButton1()
                {
                strDEL_E = "&Del_E";
                }

        </script>

        <style>
                .IO_box {
                        float: left;
                        margin: 0 20px 20px 0;
                        border: 1px solid blue;
                        padding: 0 5px 0 5px;
                        width: 390px;
                }
                h1 {
                        font-size: 240%;
                        color: black;
                        margin: 0 0 14px 0;
                }
                h2 {
                        font-size: 200%;
                        color: #5734E6;
                        margin: 5px 0 5px 0;
                }
                p, form, button {
                        font-size: 80%;
                        color: #252525;
                }
                .small_text {
                        font-size: 100%;
                        color: #737373;
                }
        </style>

    </head>

    <body onload="GetArduinoIO()">
        <h1><center>132kV Ambalangoda Line Bay 01</center></h1><h1><center>Matugam GS</center></h1>

                <div class="IO_box">
                        <h2>Equipment Status
                        <p>Line Voltage--- <span class="analog">...</span>kV</p>
                        <p>Bus1 Voltage--- <span class="analog">...</span>kV</p>
                        <p>Bus2 Voltage--- <span class="analog">...</span>kV</p>
                        <p>Busbar 01 Disconnector .. Q1 <span class="switches">...</span></p>
                        <p>Busbar 02 Disconnector .. Q2 <span class="switches">...</span></p>
                        <p>Circuit Breaker ............... Q0 <span class="switches">...</span></p>
                        <p>Line Disconnector ............ Q9 <span class="switches">...</span></p>
                        <p>Earth Switch .....................Q8 <span class="switches">...</span></p></h2
                </div>

                <div class="IO_box">
```

```
                        <h2>Alarms
                        <p>VT_MCB_TRIP-------------><span class="switches">...</span></p>
                        <p>POLE_DISCREPANCY----><span class="switches">...</span></p>
                        <p>CB_MCB_TRIP-------------><span class="switches">...</span></p>
                        <p>SPRING_DISCHARGE----><span class="switches">...</span></p>
                        <p>SYNC_NOT_OK-------------><span class="switches">...</span></p>
                        <p>INTERLOCK_NOT_OK---><span class="switches">...</span></p>
                        <p>SF6_ALARM----------------><span class="switches">...</span></p>
                        <p>SF6_LOCKOUT-------------><span class="switches">...</span></p></h2>
                    </div>

                    <div class="IO_box">
                            <h2>Event Records</h2>
                            <p></p>
                            <h1></h1>
                            <p></p>
                            <h1></h1>
                            <button type="button" id="Mem_Clear" onclick="GetButton1()">Clear Event Memory</bu
                <object data="event.txt" width="380" height="414"></object>


                    </div>

    </body></html>
```