Arduino Code:

```
// DCC Circuit Breaker Circuit
// Sketch which uses ACS712 devices to interrupt power to multiple MRR Power Districts during
short circuit conditions
// This version provides for LED tripped circuit LED indicator lights and manual buttons that
allow operator to cut power to any district
// A resistor ladder detection HW/SW combination allows multiple button presses to be
detected on a single analog input pin
// Both the display LEDs and the manual buttons are optional. Relay boards will have
integrated indicator lights if that suits your situation
const int NUMBER_OF_PDs        =  4;               // Total number of power districts to
control
const int current_board_pin[]  =  {A0, A1, A2, A3}; // analog input pins for ACS712, one per
power district
const int Relay_Pin[]          =  { 5,  4,  3,  2}; // digital pin to actuate relay, one per
power district
const int LED_Pin[]            =  { 9,  8,  7,  6}; // digital pin to light LED to indicate
tripped status on panel, one per power district
                                                    // Note: this sketch assumes bi-polar
red/green LEDs in use
// Set up manual breaker control buttons using a resistor ladder to read multiple buttons on
a single analog input pin:
#define max_counts 1023
// A/D max value for resistor ladder analogRead and current board analog pin readings
// based on #bits in microcontroller A/D. Uno is 10 bit (2^10-1 or 1023). ESP32 is 12 bit
(2^12-1 or 4095)
// analog pin from which to read multi-button resistor ladder and button matrix
= -1;            // State for manual circuit interrupt buttons
#define BUTTON_PIN A6
int ManualButtonState
int LastManualButtonState = -1;            // State for manual circuit interrupt buttons
previous time through read cycle
int ManualTripStatus[]    = {0, 0, 0, 0};  // State for manual power district off/on
selection controlled by buttons
// constants for current detection
#define arduino_operating_voltage
#define volts_per_amp
#define tripped
#define closed
const int current_board_offset_counts[] = {509, 516, 518, 502}; // (counts), ACS712 should
measure max_counts/2 at zero current; set these values by measuring analog counts at zero
current flow
// Breaker States
enum BREAKERSTATES
{
  Clear,
  Init,
  Trip,
  Recov,
// There are 4 Breaker logic states:
// Normal operation state, no trip
// State where current board first detects over-current condition; take additional current
measurements for InitTime msec to confirm trip using TripCtr/MeasCtr ratio test
// Power district has a confirmed short, wait RecovTime msec before attempting reset
// Test state after waiting RecovTime msec to see if trip is cleared; transition either to
Clear or back to Trip


};


= 0.6;


unsigned long     Now     = millis();
unsigned long ClearTime[] = {Now, Now, Now, Now}; // set time for Clear state
unsigned long TripTime[]  = {Now, Now, Now, Now}; // set time for Trip state
  5.0 // (V) nominal voltage at 5V pin of arduino, 5V pin output is used for Vcc of current
detection boards, must fall within 4.5-5.5V
0.185 // sensitivity of current board that converts amps to volts measured by Arduino - get
from board's spec sheet. 0.185 is for ACS712 +/-5amp version
    1 // relay board logic is 1 = open   (no current; tripped)
0 //                       0 = closed (circuit completed; no trip)
//Counter for measurements made during Init Phase
//Counter for number of shorts detected during Init Phase
```

```
// Time (msec) in which current measurements are made after initial short indication to
determine if circuit is really shorted
// Time (msec) to wait after circuit is tripped to attempt reset/clear
// ratio of #trip measurements to #total current measurements required for transition from
Init to Trip
// #amps for short circuit trip; set so that you can run multiple locos per power district at
high power drain but below max current sourced by command station
int MeasCtr[]
int TripCtr[]
int InitTime
int RecovTime
const float TripRatio
const float TripCurrent = 1.8;
const int TripCounts  = TripCurrent * volts_per_amp * (float)max_counts /
arduino_operating_voltage; // This is the #counts above or below the nominal A/D count
reading that = TripCurrent
= {0, 0, 0, 0};
= {0, 0, 0, 0};
= 120;
= 3000;
BREAKERSTATES BreakerState[] = {Clear, Clear, Clear, Clear}; // initial Breaker state
void setup() {
  for (int disNum = 0; disNum<NUMBER_OF_PDs; disNum++) {
    pinMode(BUTTON_PIN,                INPUT);
    pinMode(current_board_pin[disNum], INPUT);
    pinMode(Relay_Pin[disNum],        OUTPUT);
    pinMode(LED_Pin[disNum],          OUTPUT);
    digitalWrite(Relay_Pin[disNum],   closed);  // startup state for circuit breaker is
closed (i.e. power is on to each PD)
    digitalWrite(LED_Pin[disNum],     HIGH  );  // startup indication on LED is for green
with HIGH on LED_Pin
}
  Serial.begin(115200);   // Un-comment if using diagnostic print commands below
}
void loop() {


//


LastManualButtonState  = ManualButtonState;
ManualButtonState      = readAnalogButton();  // return value is -1 for no press, 0-3 =
button 1-4 pressed
    Serial.println(ManualButtonState);               // Debug statement as needed
  if ( ManualButtonState > LastManualButtonState ) {  // Only toggle on rising edge of button
reading
    digitalWrite(Relay_Pin[ManualButtonState],!digitalRead(Relay_Pin[ManualButtonState])); //
Toggle relay to change manual disctrict power off/on
    digitalWrite(LED_Pin[ManualButtonState],  !digitalRead(LED_Pin[ManualButtonState]  )); //
Toggle indicator LED to show off/on state
    ManualTripStatus[ManualButtonState]     = !ManualTripStatus[ManualButtonState]       ; //
Set state variable ignores current detection breaker logic for this district if manual button
is set "tripped"
    }
 for (int disNum = 0; disNum<NUMBER_OF_PDs; disNum++) {
  if( ManualTripStatus[disNum] ) break; //break out of "disNum for loop" on any Manual Trip
districts
//Now check for overcurrent
  int ReadCounts = analogRead(current_board_pin[disNum]) -
current_board_offset_counts[disNum]; // yields count value to compare to TripCounts value;
// Some useful diagnostic print statements for serial monitor or plotter:
// Serial.println(analogRead(current_board_pin[0])); // measure each AC712 board separately
by reading current_board_pin[0] through current_board_pin[N]
// Serial.println(abs(ReadCounts));                    // Uncomment this to see ReadCounts
for all boards
// if(TripCtr[1] >0)  Serial.println("MeasCtr: " + String (MeasCtr[1]) + "TripCtr: " + String
(TripCtr[1]));  // Print out optional
  switch(BreakerState[disNum]) // loop through each district as directed by BreakerState
  {
    case Clear:
       if ( abs(ReadCounts) > TripCounts ) {BreakerState[disNum] = Init; }   // Current limit
exceeded so initialize trip; abs() accounts for minus current readings in DCC


else break;


{ClearTime[disNum] = millis();}   // No Trip, so save time of currecnt Clear state
```

```
// During trip initialization, sample current readings for duration InitTime, then test if
ratio of trips/measurements exceeds threshold
// Measurements are counted each time through InitTime duration
case Init:
   MeasCtr[disNum]++;
   if (abs(ReadCounts) > TripCounts ) TripCtr[disNum]++;    // Increment TripCtr if trip
current is exceeded on this measurement
   if ( (millis() - ClearTime[disNum] ) > InitTime ){  // Wait InitTime msec to determine if
circuit is tripped
if ( (float)TripCtr[disNum]/(float)MeasCtr[disNum] > TripRatio ) { BreakerState[disNum] =
Trip;  TripTime[disNum]  = millis();} // The ratio test for #trips/#meas is met so circuit is
tripped
else { TripCtr[disNum] = 0; MeasCtr[disNum] = 0; BreakerState[disNum] = Clear;
ClearTime[disNum] = millis();}                   // If ratio test fails, assume a false trip
and go back to Clear state


} }


}


} break;


case Trip:
   digitalWrite(Relay_Pin[disNum], tripped);    // Disable relay since circuit is tripped
   digitalWrite(LED_Pin[disNum],   LOW    );    // Change LED_pin since circuit is tripped
   if ( (millis() - TripTime[disNum]) > RecovTime ){ BreakerState[disNum] = Recov;
TripCtr[disNum] = 0; MeasCtr[disNum] = 0;}  // Once RecovTime msec have elapsed, Reset
counters for
break;
// next test through switch logic,& try to Recover from trip
case Recov:
   digitalWrite(Relay_Pin[disNum], closed);  // Enable relay for current sense test
   digitalWrite(LED_Pin[disNum],   HIGH  );  // Set LED_Pin  for current sense test
   int ReadCounts = analogRead(current_board_pin[disNum]) -
current_board_offset_counts[disNum]; // yields count value to compare to test value; abs()
accounts for minus current readings in DCC (voltage < offset)
   if ( abs(ReadCounts) > TripCounts ) {BreakerState[disNum] = Trip;   TripTime[disNum] =
millis();}  // if still overcurrent, transition back to Trip State
   else                          {BreakerState[disNum] = Clear; ClearTime[disNum] =
millis();}  // else trip has been Cleared


break;


  // End Breaker switch statement
  // End power district "for" loop
  // End main loop
// Function to read set of buttons on analog input pin for circit breaker manual on/off
    •  //     (-1 = no press, 0-3 = button 1-4 pressed)
    •
    •  //     Reference: http://www.ignorantofthings.com/2018/07/the-perfect-multi-button-
       input-resistor.html
    •  int readAnalogButton() {
    •    float avg = max_counts / float(NUMBER_OF_PDs);
    •


  int   val = analogRead(BUTTON_PIN);
  if ( val > (NUMBER_OF_PDs - 0.5) * avg ) return -1;
  for (int i = 0; i < NUMBER_OF_PDs; i++) {
    if ( val < round((i + 0.5) * avg) )    return  i; }


}
```