SCADA CONTROL

2018 Fall –2019 Spring

APRIL 8, 2019

Table of Contents

Summary

We SCADA CONTROL team are making the control of motor automatically. Our solution is using the stepper motor and HMI to improve it. The original function can only be used by manual control, and we can control it by using the HMI now. It is more convenient for the users to control, which also benefits the ECE labs.

Background

The UI Department of Electrical and Computer Engineering laboratory facilities include an analog model power system (AMPS) that is capable of simulating interaction of control and protection hardware in a network with up to five lines transmission line segments. The system protection hosts a full complement of commercial protective relays and a fault generator capable of initiating common fault types with any fault impedance and any duration. Multiple generation sources can be interfaced with the system including synchronous machines, a doubly fed induction generator and power electronically coupled generation. The facility shown in Figure is used for labs for several power systems courses as well as for research projects.

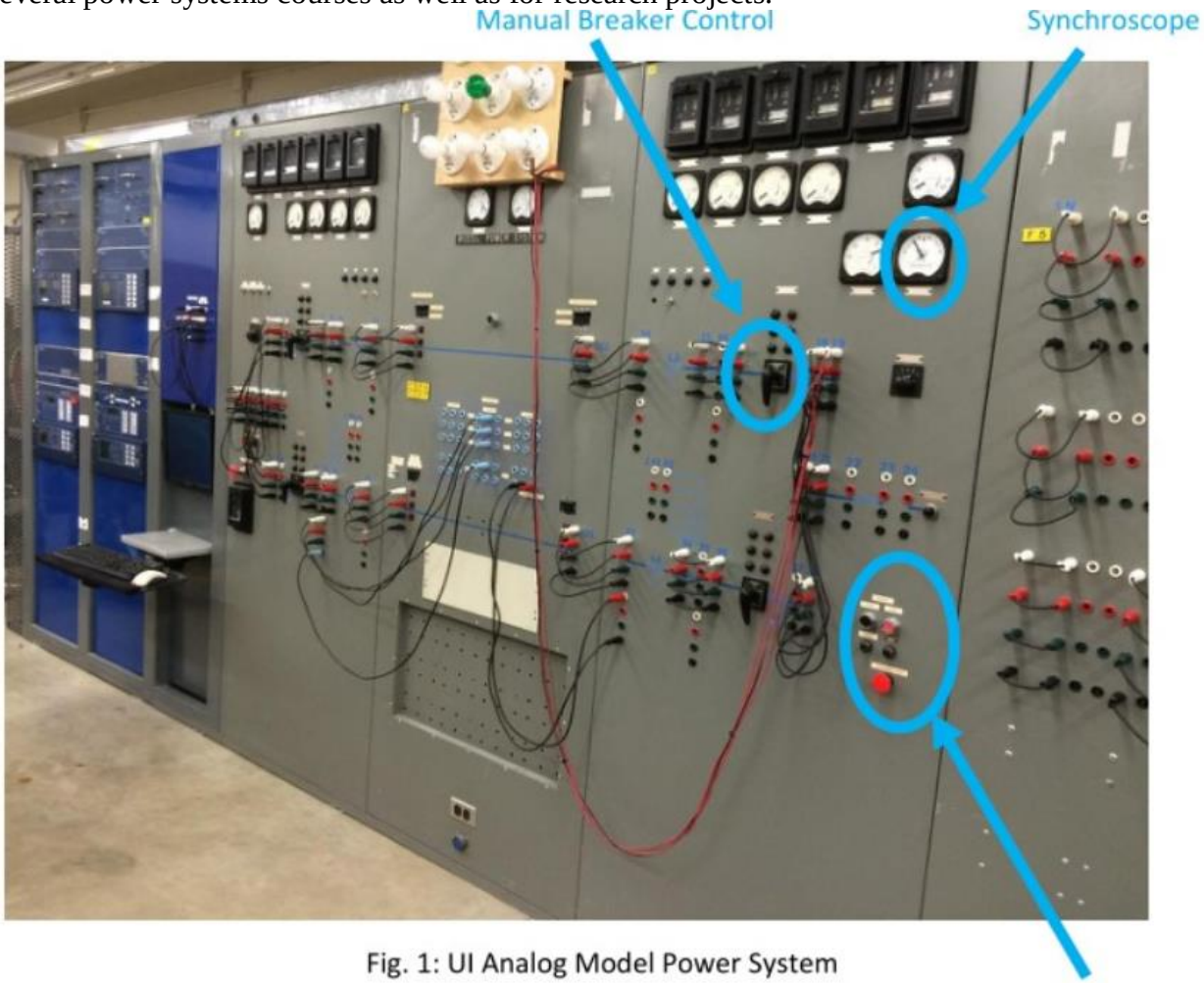


Fig. 1: UI Analog Model Power System

The model power system has controls to control a synchronous generator for manual synchronization to the Avista system through the building power supply. The manual controls include push buttons for start and start and potentiometers to control speed and terminal voltage as shown in Figure 1. The potentiometers regulator a dc control signals for the MG set. There is also a synchroscope for visual indication to determine when to manually close the circuit breaker. While the circuit breaker can be remotely controlled through a supervisory control and automation (SCADA) interface, the machine itself cannot be controlled through SCADA.

Problem Definition

- Learning the principle and operation of Synchronous motor generator and SCADA Control system.
- Using PC software (which is little relative to the Programmable Logic Controller) to control the rotation speed of the generator.
- Being able to adjust speed and frequency of the motor, showing the rotate frequency of the Synchronous motor in the scope successfully.

Project Members

- Kevin (Team leader) is the primary client contact and in charge of budget
- Ziang is wiki master and keeps team documentation
- Hui will organize team meetings

Concepts Considered

Final four proposals:

1. Stepper Motor: Connect stepper motors with button "Speed" and "Voltage", type code to control the stepper motors.
2. ABB: ABB's industrial robot controllers offer superior motion control and enable quick integration of additional hardware.
3. DC Voltage Supply: A DC power supply is one that supplies a constant DC voltage to its load. Depending on its design, a DC power supply may be powered from a DC source or from an AC source such as the power mains.
4. Digital Potential Meter: Digital potentiometers are used to trim and scale analog signals, which provide an output resistance that is variable based on digital inputs and so are sometimes referred to as resistive digital-to-analog converters

Concept Selection

FINAL CHOICE: Step motor and HMI

- Step motor:
    1. Stable. Can drive a wide range of frictional and inertial loads.
    2. Needs no feedback. The motor is also the position transducer.
    3. Inexpensive relative to other motion control systems.
    4. Standardized frame size and performance.
    5. Plug and play. Easy to setup and use.
    6. Safe. If anything breaks, the motor stops.
    7. Long life. Bearings are the only wear-out mechanism.
    8. Excellent low speed torque. Can drive many loads without gearing.
    9. Excellent repeatability. Returns to the same location accurately.
    10. Overload safe. Motor cannot be damaged by mechanical overload.

Installing HMI into 3530 RTAC

- HMI:
    1. Deliver efficient HMI solutions by building effective operator interface screens through easy-to-use tools without the need for mapping data tags.
    2. Identify changing system conditions and make informed decisions based on real-time analysis without the need of any special software to view the HMI.
    3. Leverage powerful and reliable RTAC hardware with a cost-effective HMI option for local/remote monitoring, control, integrated alarms, and annunciation.
    4. Monitor your system and analyze performance anywhere, anytime with a secure, web-based, thin client user interface, which Including morphological charts, decision matrices as appropriate
    5. Access the RTAC HMI locally or remotely via a web browser interface from the web server on the RTAC unit. On demand visualization and control makes monitoring and controlling your system a more efficient task. Role-based accounts provide appropriate security access. Since it is a thin client, no installation and no upkeep for a specific application are required.
    6. The RTAC HMI offers an easy way to visualize data and create custom diagrams to monitor and control your system. The HMI provides authenticated access for multiple users from multiple locations and is viewable from a web browser. It renders natively on browsers compatible with HTML5 web standard—no plugins required.

- The HMI creation explanations:

    From the above is the steps of the Human Machine Interface creating process. We depend on the SEL devices (SEL 3530, SEL 2440, SEL 734, SEL 351 and the computer center), and I will explain every device's working principle and the aims when we did in the past time of using in our project.

    SEL 3530 which is RTAC (real time automatic control), every remote bite is controlled by this device. Such as the SEL734, SEL 351S. The SEL-3530 Real-Time Automation Controller (RTAC) provides complete and flexible system control with integrated security, seamless configuration, unified logic, and reliability. The RTAC converts data between multiple protocols, communicates with any configured and connected device.

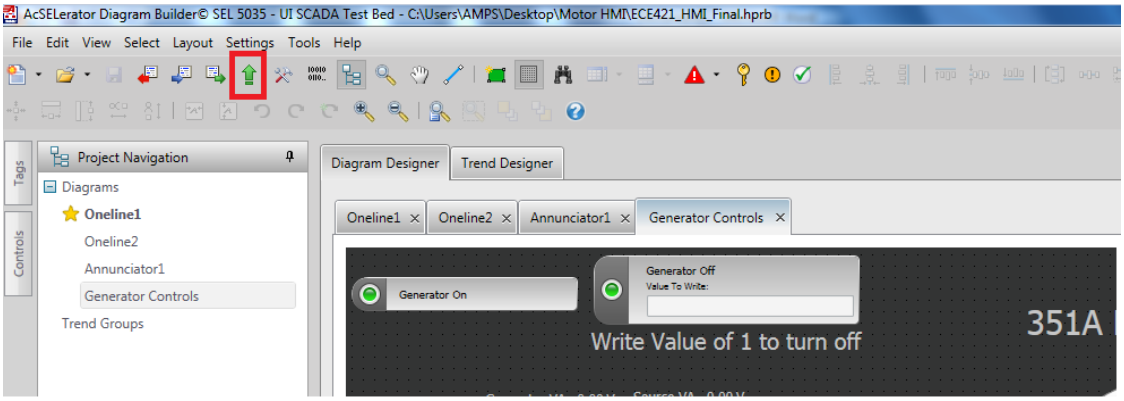
    SEL 734 which is the electrical meter. We use ACSELERATOR QuickSet to customize your metering. Set and edit meter configuration, settings, and logic. And as the working of the SEL 734 in the project:
    • Current, voltage, power, frequency, supply voltage interruptions, dips and swells, harmonics, inter harmonics, flicker, and unbalance
    • Measurement aggregation
    • Harmonic phase angles for voltage and current

• High-resolution, 512-samples/cycle waveform capture
• Total harmonic distortion (THD), crest factor, and K-factor metering with up to 63rd harmonic content
• High-speed load profile recording with three-second resolution
• Symmetrical components (unbalance)
• Real-time waveforms with the Wave view oscillography functionality.

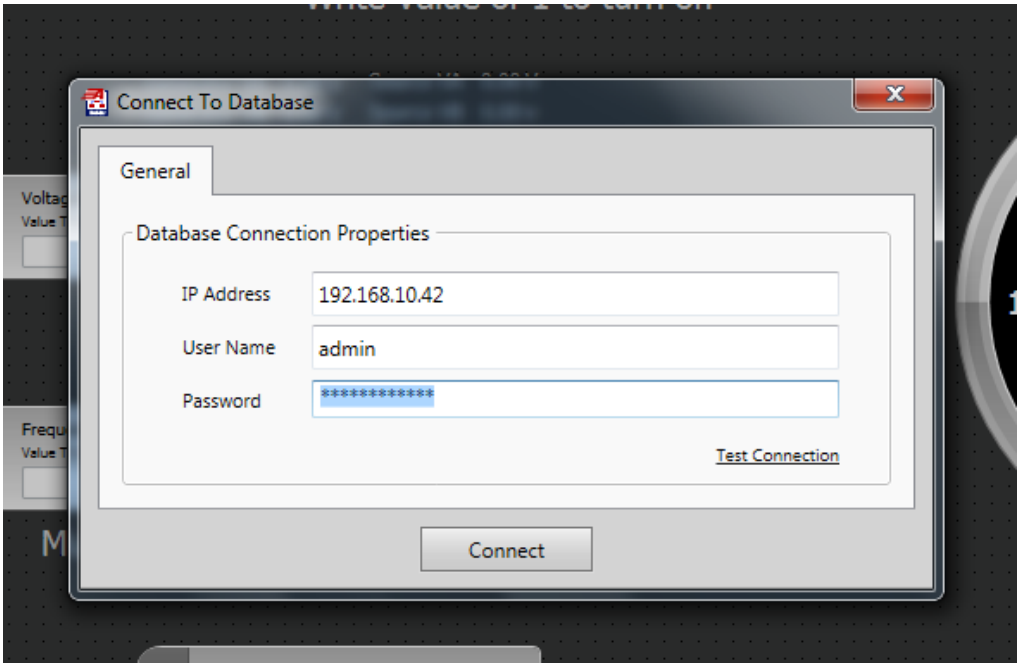
SEL 2440. In the process of the new HMI platform designing. We depends on the SEL 2440 to distinguish the condition of the breaker. The SEL-2440 DPAC is a 48-point discrete programmable automation controller ideal for utility. And it is the good communicator to connect the controller and the power system. And it is easy to maintain and support, and it meets stringent protective relay standards. Available mounting options to meet our application needs include rack, panel, surface, and DIN mounts.

SEL 351S which is used to control the breaker's closing and opening. Reliable Breaker Control—Open or close the circuit breaker manually with the optional Safe Lock trip/close pushbuttons, which provide direct control of the breaker independent of the relay. Reclosing Control—Program the relay four-shot auto reclosing function with synchronism and voltage check logic to mimic a variety of reclosing practices. Sequence coordination logic coordinates with downstream reclosers for a complete protection scheme.

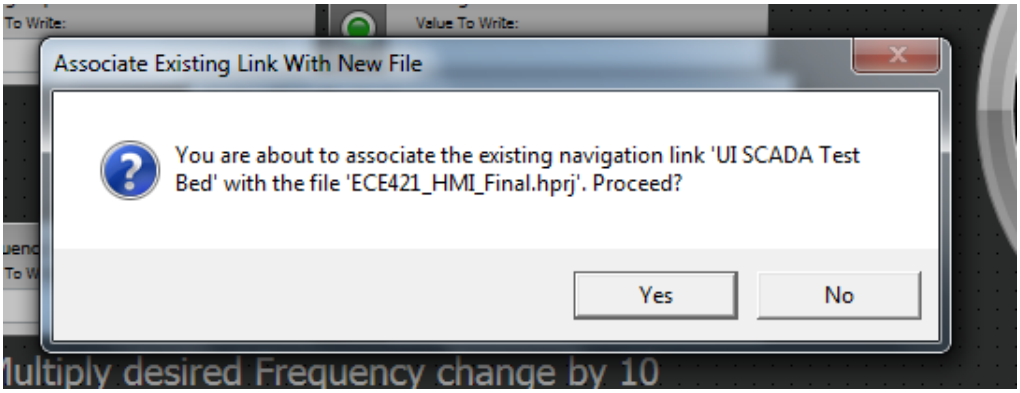
Step 1. Open the Diagram Builder Software from the windows start menu
Step 2. Click the File button and Open Project
Step3. Locate the ECE421_HMI_Final.hprb file found at C:\Users\AMPS\Desktop\Motor HMI
Step 4. Open the file and click on the green arrow located in the ribbon in Diagram Builder to send the HMI to the 3530 RTAC.



Step 5. Login to RTAC in Diagram Builder
IP Address: 192.168.10.42
User Name: admin
Password: ModelPower_1
Click connect



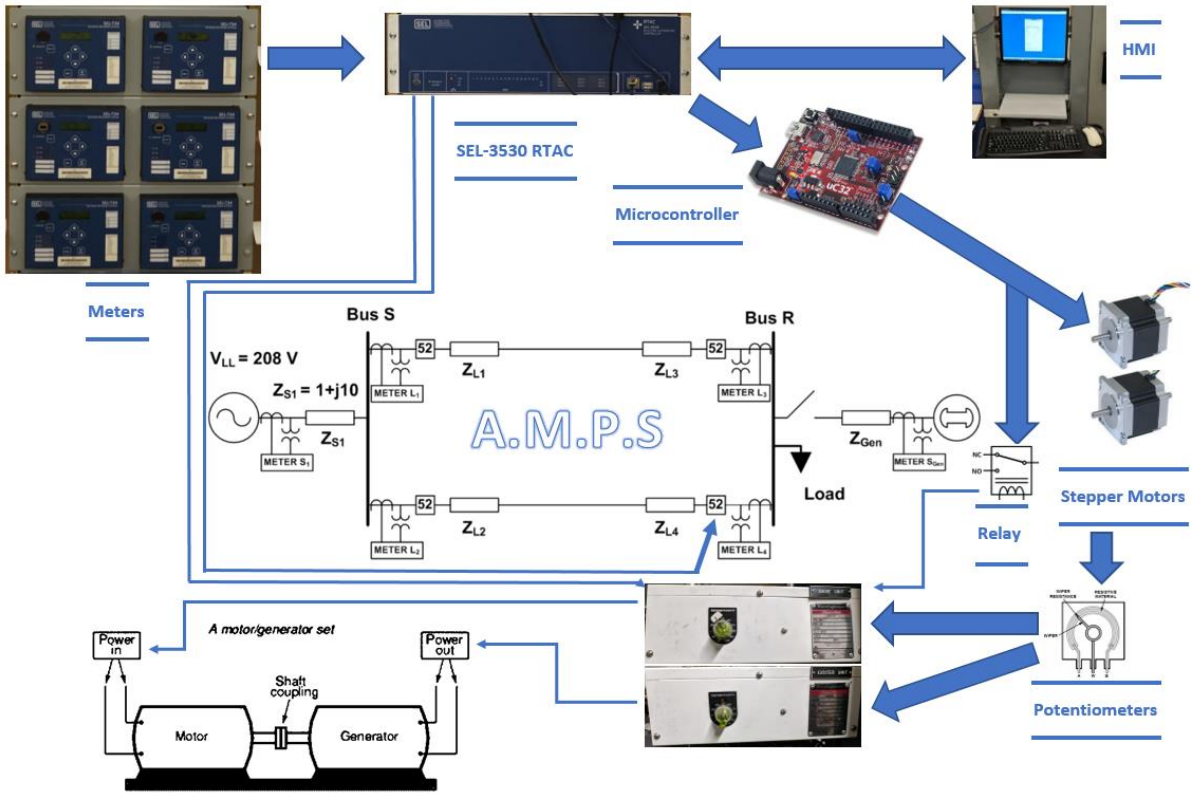
Click Yes to send HMI to RTAC



The HMI is now uploaded to the RTAC and can be viewed in the web browser. Click on HMI project on the bottom left of the web browser menu to access.

System Architecture

A rough layout of how everything is connected can be seen in the figure below.



- Start/Stop

  Start and stop for the motor/generator set is currently controlled by two push buttons. The start pushbutton is in a normally closed position. A wire was connected to a contact on each side of the pushbutton, and these wires were then connected to a normally open relay on the back of the RTAC. On the HMI, when the Generator On button is selected, it sends a pulse to this relay which temporarily closes it. This turns on the generator.

  The pushbutton that stops the generator is in a normally closed position. To turn the generator off, a mechanical relay was placed in series with the pushbutton. This relay is also normally closed to allow for the generator to run even if power to the microcontroller that is connected to the relay is lost. On the HMI, when a 1 command is sent to the microcontroller from the Generator Off button, it received by the microcontroller. The microcontroller communicates with the HMI through the RTAC using MODBUS protocol. Once, the off command is received, the microcontroller opens the relay and shuts off the generator.

- Voltage/Frequency Control

  Voltage and frequency adjustments are controlled by adjusting potentiometers. To add remote control, dual shafted stepper motors were used. One shaft connected to the

potentiometer, and the other shaft connected to the potentiometers old knob. This allowed for both remote and manual control. The stepper motors are controlled by the microcontroller. The microcontroller receives either a voltage up, voltage down, frequency up, or frequency down value for controlling the voltage and frequency.

- Breaker

  Breaker 4 is the breaker being used to close the generator in with the grid. On the underside of this breaker, there are two rows of bolts for connecting wires. On the left row, the two front bolts can be used to open and close the breaker. This is done by shorting that respective bolt with any of the bolts with wires connected to them on the right-hand side. The front bolt on the left side closes the breaker, while the second one back opens it.

  Three wires were run from these bolts to relays on the RTAC. The open/close wires were connected to separate breakers. The third wire, the one they short too, was connected to the other side of those breakers. In the HMI, pressing the Breaker Open or Breaker Close buttons will send a pulse to that respective relay and will open or close the breaker.

Design Evaluation

In the design validation plan, 10 requirements were listed. They also had tests listed with them. They can be seen in the table below. The requirements are on the left, and the test is on the right.

Speed of generator (Frequency) can be controlled from computer	Change speed from computer and see if frequency dial has changed
Terminal voltage of generator can be controlled from computer	Change voltage on computer and see if voltmeter has changed
Start of generator can be controlled from computer	Press start on computer and see if generator starts
Stop of generator can be controlled from computer	With generator running, press stop on computer and see if generator stops
View generator terminal voltage on computer	Check voltage and computer and compare to voltmeter on panel
View generator frequency on computer	Check frequency on computer and compare to frequency on panel
View synchroscope on computer	Compare computer and panel synchroscopes
Close and open breakers from computer	Send command from computer and verify breaker has open/closed
(Time Permitting) Maintain all manual controls currently on panels	Use controls on panel and verify proper changes come from these
(Time Permitting) Press single button on computer to auto synch voltage, frequency, and close breaker at proper time	Press button on computer and watch change in voltage, frequency, and closing of breaker to tie generator into grid

The speed of generator can be adjusted by sending a command to adjust the frequency up, or the frequency down. Per the test, the frequency dial can be seen moving. Furthermore, the measured frequency can be seen to change. Sending a command to change the voltage up, or down can also be measured by observing the changes in the voltmeter. These changes can also be seen from the measured voltages through the HMI.

The generator start button can be seen to be working by noting that the generator starts after pushing the generator on button. Similarly, the generator will turn off after pressing the generator off button.

The voltage and frequency measured through the HMI can be checked against the analog voltage and frequency meters on the analog model power system. They show similar values. Additionally, the synchroscope can be observed on the HMI to spin in a similar manner as to the manual synchroscope. Albeit, there is a small delay on the HMI based synchroscope.

Opening and closing the breaker can be verified after sending the commands from the HMI by seeing the breaker open and close. All manual controls were maintained. At this time, an autosync function does not work though.

Future Work

- Autosync
  This will require bringing in an expert with experience in either the SEL 351s or the SEL 420. If they can troubleshoot the synchronism check error and fix it, then implementing the autosync can be accomplished in less than a days worth of work. It will require doing a synchronism check, if it passes, then send command to close breaker. If it fails, then logic will adjust the voltage/frequency to within an acceptable range and rerun a synchronism check.
- Remove HMI Synchroscope delay
  This will also require in depth investigation, or an expert, to determine what is causing the lag on the digital synchroscope. Once the lag has been determined, then a work around can be put in place.

Appendices

Wiki page: http://mindworks.shoutwiki.com/wiki/SCADA_Control

Microcontroller Code

```
#include <ModbusRtu.h>

// data array for modbus network sharing
uint16_t au16data[16] = {
  0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, -1 };
```

```
//This sets up the MODBUS
Modbus slave(1,0,0); // this is slave @1 and RS-232 or USB-FTDI

void setup() {
 slave.begin( 19200 ); // baud-rate at 19200 for modbus
 pinMode(72, OUTPUT); //sets pin as output for stepper motor
 pinMode(73, OUTPUT); //sets pin as output for stepper motor
 pinMode(74, OUTPUT); //sets pin as output for stepper motor
 pinMode(75, OUTPUT); //sets pin as output for stepper motor
 pinMode(76, OUTPUT); //sets pin as output for stepper motor
 pinMode(77, OUTPUT); //sets pin as output for stepper motor
 pinMode(78, OUTPUT); //sets pin as output for stepper motor
 pinMode(79, OUTPUT); //sets pin as output for stepper motor
 pinMode(80, OUTPUT); //sets pin as output for relay (generator off)

 digitalWrite(73, LOW); //ensures pin is set to low so that stepper motor can be manually spun
 digitalWrite(75, LOW); //ensures pin is set to low so that stepper motor can be manually spun
 digitalWrite(72, LOW); //ensures pin is set to low so that stepper motor can be manually spun
 digitalWrite(74, LOW); //ensures pin is set to low so that stepper motor can be manually spun
 digitalWrite(77, LOW); //ensures pin is set to low so that stepper motor can be manually spun
 digitalWrite(79, LOW); //ensures pin is set to low so that stepper motor can be manually spun
 digitalWrite(76, LOW); //ensures pin is set to low so that stepper motor can be manually spun
 digitalWrite(78, LOW); //ensures pin is set to low so that stepper motor can be manually spun
 digitalWrite(80, LOW); //ensures pin is set to low so that generator doesn't get turned off on accident
}

void loop() {
 int Vup = 0; //integer for copying positive voltage change commands
 int Vdown = 0; //integer for copying negative voltage change commands
 int Fup = 0; //integer for copying positive frequency change commands
 int Fdown = 0; //integer for copying negative frequency change commands


 while(1){
  slave.poll( au16data, 16 ); //polls modbus registers
  if(au16data[6]!=0){ //checks if stop! command has been sent and then clears out rest of registers if it
has been
    au16data[0] = 0;
    au16data[1] = 0;
    au16data[2] = 0;
    au16data[3] = 0;
    au16data[4] = 0;
    au16data[5] = 0;
  }
  /*if (au16data[0]==1){
    GenOn();
    au16data[0]=0;
```

```
   }*/
   //This code removed because Gen On was done with RTAC
   if (au16data[1]==1){ //checks for generator off command
    GenOff();
    au16data[1]=0;
   }
   if(au16data[2]!=0){ //checks for voltage up value
    Vup = au16data[2]; //sets Vup value for voltage up change
    VStepperControl(1, Vup); //calls funtion for stepper motor and passes direction and voltage change
value
    au16data[2] = 0; //clears voltage up change MODBUS register
   }
   if(au16data[3]!=0){ //checks for voltage down value
    Vdown = au16data[3]; //sets Vdown value for voltage down change
    VStepperControl(0, Vdown); //calls funtion for stepper motor and passes direction and voltage change
value
    au16data[3] = 0; //clears voltage down change MODBUS register
   }
   if(au16data[4]!=0){ //checks for frequency up value
    Fup = au16data[4]; //sets Fup value for frequency up change
    FStepperControl(1, Fup); //calls funtion for stepper motor and passes direction and frequency change
value
    au16data[4] = 0; //clears frequency up change MODBUS register
   }
   if(au16data[5]!=0){ //checks for frequency down value
    Fdown = au16data[5]; //sets Fdown value for frequency down change
    FStepperControl(0, Fdown); //calls funtion for stepper motor and passes direction and frequency
change value
    au16data[5] = 0; //clears frequency down change MODBUS register
   }
  }
}

//function for controlling the stepper motor that controls voltage
void VStepperControl(int VDir, int VoltValue){
 int VDelay;
 int VC = 3; //voltage constant. change this to change the rate the voltage change value is amplified by
 int turn = VC*VoltValue;
 if (VoltValue <= 5){
  VDelay = 25;
 }
 else if (VoltValue <= 10){
  VDelay = 20;
 }
 else if (VoltValue <= 15){
  VDelay = 15;
 }
 else if (VoltValue <= 20){
```

```
  VDelay = 10;
}
else if (VoltValue <= 25){
  VDelay = 5;
}
else{
  turn = VC*25;
  VDelay = 5;
}
for (int i=turn; i>0; i--){
  slave.poll( au16data, 16 );
  if(au16data[6]!=0){
    au16data[0] = 0;
    au16data[1] = 0;
    au16data[2] = 0;
    au16data[3] = 0;
    au16data[4] = 0;
    au16data[5] = 0;
    return;
  }
  if (VDir == 1){
    digitalWrite(73, LOW);
    digitalWrite(75, LOW);
    digitalWrite(72, HIGH);
    digitalWrite(74, HIGH);
    delay(VDelay);
    digitalWrite(72, LOW);
    digitalWrite(75, LOW);
    digitalWrite(73, HIGH);
    digitalWrite(74, HIGH);
    delay(VDelay);
    digitalWrite(72, LOW);
    digitalWrite(74, LOW);
    digitalWrite(73, HIGH);
    digitalWrite(75, HIGH);
    delay(VDelay);
    digitalWrite(73, LOW);
    digitalWrite(74, LOW);
    digitalWrite(72, HIGH);
    digitalWrite(75, HIGH);
    delay(VDelay);
  }
  else{
    digitalWrite(73, LOW);
    digitalWrite(74, LOW);
    digitalWrite(72, HIGH);
    digitalWrite(75, HIGH);
    delay(VDelay);
```

```
    digitalWrite(72, LOW);
    digitalWrite(74, LOW);
    digitalWrite(73, HIGH);
    digitalWrite(75, HIGH);
    delay(VDelay);
    digitalWrite(72, LOW);
    digitalWrite(75, LOW);
    digitalWrite(73, HIGH);
    digitalWrite(74, HIGH);
    delay(VDelay);
    digitalWrite(73, LOW);
    digitalWrite(75, LOW);
    digitalWrite(72, HIGH);
    digitalWrite(74, HIGH);
    delay(VDelay);
   }
  }
  digitalWrite(73, LOW);
  digitalWrite(75, LOW);
  digitalWrite(72, LOW);
  digitalWrite(74, LOW);
  return;
}

//function for controlling the stepper motor that controls frequency
void FStepperControl(int FDir, int FreqValue){
  int FDelay;
  int FC = 6;  //frequency constant. change this to change the rate the frequency change value is amplified
by
  int turn = FC*FreqValue;
  if (FreqValue <= 10){
    FDelay = 25;
  }
  else if (FreqValue <= 20){
    FDelay = 20;
  }
  else if (FreqValue <= 30){
    FDelay = 15;
  }
  else if (FreqValue <= 40){
    FDelay = 10;
  }
  else if (FreqValue <= 50){
    FDelay = 5;
  }
  else{
    turn = FC*50;
    FDelay = 5;
```

```
}
for (int i=turn; i>0; i--){
 slave.poll( au16data, 16 );
 if(au16data[6]!=0){
  au16data[0] = 0;
  au16data[1] = 0;
  au16data[2] = 0;
  au16data[3] = 0;
  au16data[4] = 0;
  au16data[5] = 0;
  return;
 }
 if (FDir == 1){
  digitalWrite(77, LOW);
  digitalWrite(79, LOW);
  digitalWrite(76, HIGH);
  digitalWrite(78, HIGH);
  delay(FDelay);
  digitalWrite(76, LOW);
  digitalWrite(79, LOW);
  digitalWrite(77, HIGH);
  digitalWrite(78, HIGH);
  delay(FDelay);
  digitalWrite(76, LOW);
  digitalWrite(78, LOW);
  digitalWrite(77, HIGH);
  digitalWrite(79, HIGH);
  delay(FDelay);
  digitalWrite(77, LOW);
  digitalWrite(78, LOW);
  digitalWrite(76, HIGH);
  digitalWrite(79, HIGH);
  delay(FDelay);
 }
 else{
  digitalWrite(77, LOW);
  digitalWrite(78, LOW);
  digitalWrite(76, HIGH);
  digitalWrite(79, HIGH);
  delay(FDelay);
  digitalWrite(76, LOW);
  digitalWrite(78, LOW);
  digitalWrite(77, HIGH);
  digitalWrite(79, HIGH);
  delay(FDelay);
  digitalWrite(76, LOW);
  digitalWrite(79, LOW);
  digitalWrite(77, HIGH);
```

```
    digitalWrite(78, HIGH);
    delay(FDelay);
    digitalWrite(77, LOW);
    digitalWrite(79, LOW);
    digitalWrite(76, HIGH);
    digitalWrite(78, HIGH);
    delay(FDelay);
   }
  }
  digitalWrite(77, LOW);
  digitalWrite(79, LOW);
  digitalWrite(76, LOW);
  digitalWrite(78, LOW);
  return;
}

/*void GenOn(void){
  delay(250);
  digitalWrite(80, HIGH);
  delay(2000);
  digitalWrite(80, LOW);
  delay(250);
  return;
}*/
//This code removed because Generator On was sent through the RTAC

//this code turns off the generator
void GenOff(void){
  delay(250);
  digitalWrite(80, HIGH);
  delay(4000);
  digitalWrite(80, LOW);
  delay(250);
  return;
}
```

Microcontroller Pinout