Control Rod Drive MG

Simulator

The intent of this simulator is to provide the student a hands on operation of the Control Rod Drive Mechanism Motor Generators. Using the MP E-21.6 procedure. It comprises of switches and indications to best simulate actual plant equipment .All controlled via a microcontroller and related software.

Greever, James
Pacific Gas and Electric
1/28/2016

Panel meters that indicate true values

Digital Meter that indicates true values

Volts Adj

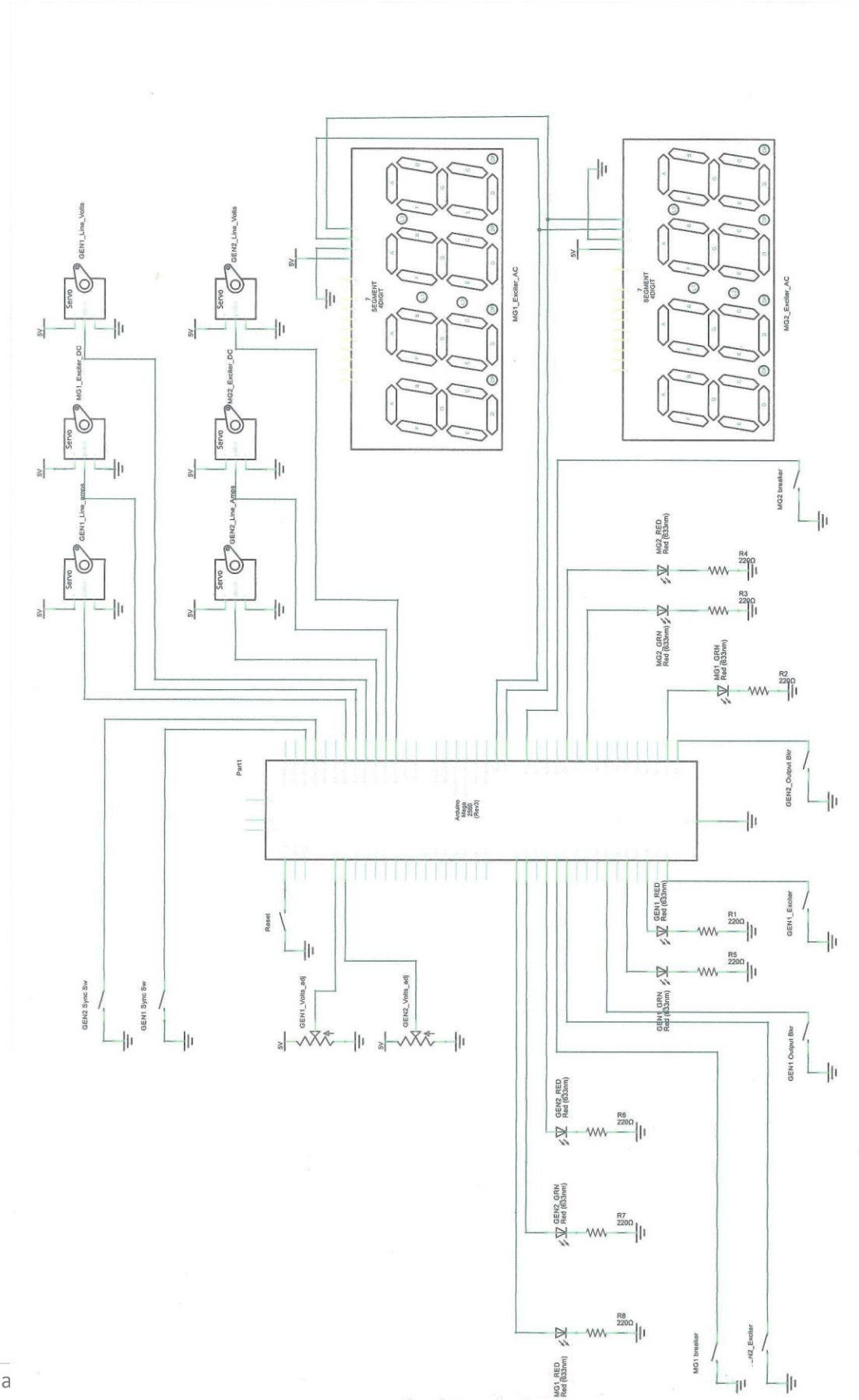
Exciter field flash pushbutton

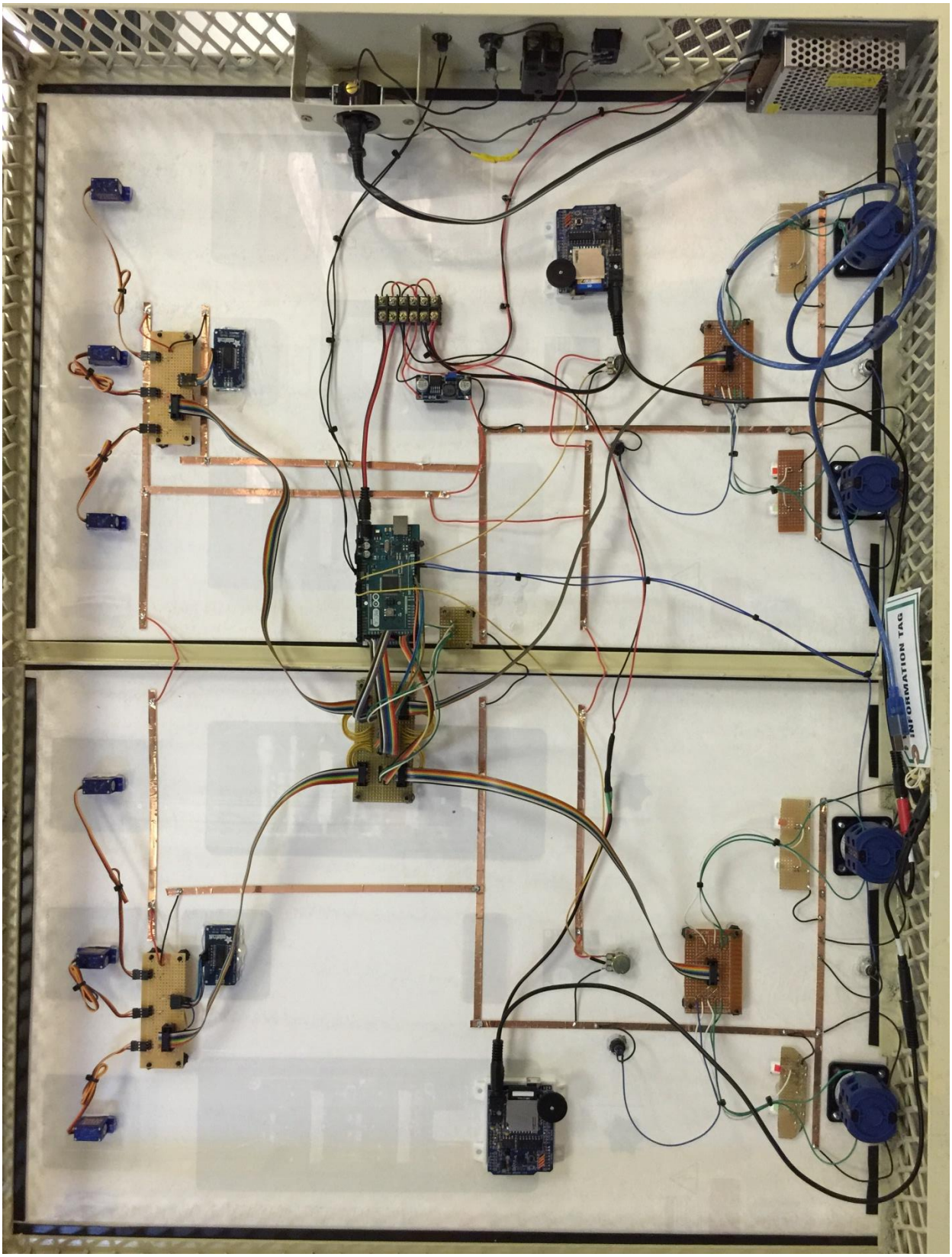
Motor operating sounds

Breaker Open / Close indication

Breaker Open / Close switches

Sync Switch

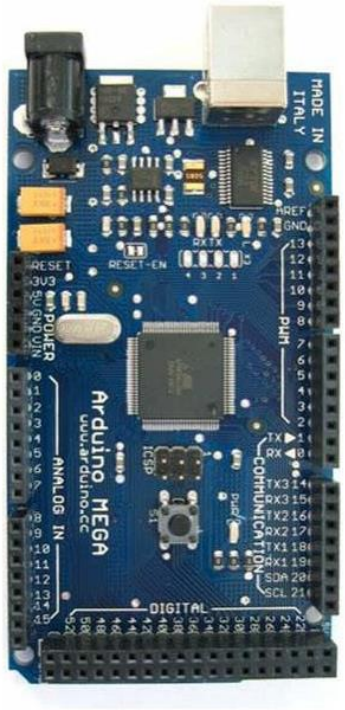

GEN1_Line_Volts
GEN2_Line_Volts
MG1_Exciter_DC
MG2_Exciter_DC
GEN1_Line_Amps
GEN2_Line_Amps
Servo
5V
7 SEGMENT 4DIGIT
MG1_Exciter_AC
MG2_Exciter_AC
MG2 breaker
MG2_RED Red (633nm)
MG2_GRN Red (633nm)
MG1_GRN Red (633nm)
R4 220Ω
R3 220Ω
R2 220Ω
Part1
Arduino Mega 2560 (Rev3)
GEN2_Output Blr
Reset
GEN1_Volts_adj
GEN2_Volts_adj
GEN2 Sync Sw
GEN1 Sync Sw
GEN1_RED Red (633nm)
GEN1_GRN Red (633nm)
R1 220Ω
R5 220Ω
GEN1_Exciter
GEN1 Output Blr
GEN2_RED Red (633nm)
GEN2_GRN Red (633nm)
R6 220Ω
R7 220Ω
R8 220Ω
MG1_RED Red (633nm)
MG1 breaker
_N2_Exciter


INFORMATION TAG

Name	GEN 1	GEN 2
Volts Adj	A0	A1
Exciter	53	43
Sync sw	11	10
Mg Bkr (close)	42	22
Gen Bkr (close)	47	37
Mg Bkr (trip)	40	24
Gen Bkr (trip)	45	35
Mg Red led	38	26
Mg Grn led	36	28
Gen Red led	51	41
Gen Grn led	49	39
Mg Gen Volts	5	2
AC amps	6	3
Gen Amps out	7	4
DC amps	Rx / Tx	Rx / Tx
Sound	12	13

ANALOG IN		
ARDUINO	ATmega 1280	
PIN 0	PIN 97	PF0 (ADC0)
PIN 1	PIN 96	PF1 (ADC1)
PIN 2	PIN 95	PF2 (ADC2)
PIN 3	PIN 94	PF3 (ADC3)
PIN 4	PIN 93	PF4 (ADC4/TCK)
PIN 5	PIN 92	PF5 (ADC5/TMS)
PIN 6	PIN 91	PF6 (ADC6/TDO)
PIN 7	PIN 90	PF7 (ADC7/TDI)

PIN 8	PIN 89	PK0 (ADC8/PCINT16)
PIN 9	PIN 88	PK1 (ADC9/PCINT17)
PIN 10	PIN 87	PK2 (ADC10/PCINT18)
PIN 11	PIN 86	PK3 (ADC11/PCINT19)
PIN 12	PIN 85	PK4 (ADC12/PCINT20)
PIN 13	PIN 84	PK5 (ADC13/PCINT21)
PIN 14	PIN 83	PK6 (ADC14/PCINT22)
PIN 15	PIN 82	PK7 (ADC15/PCINT23)



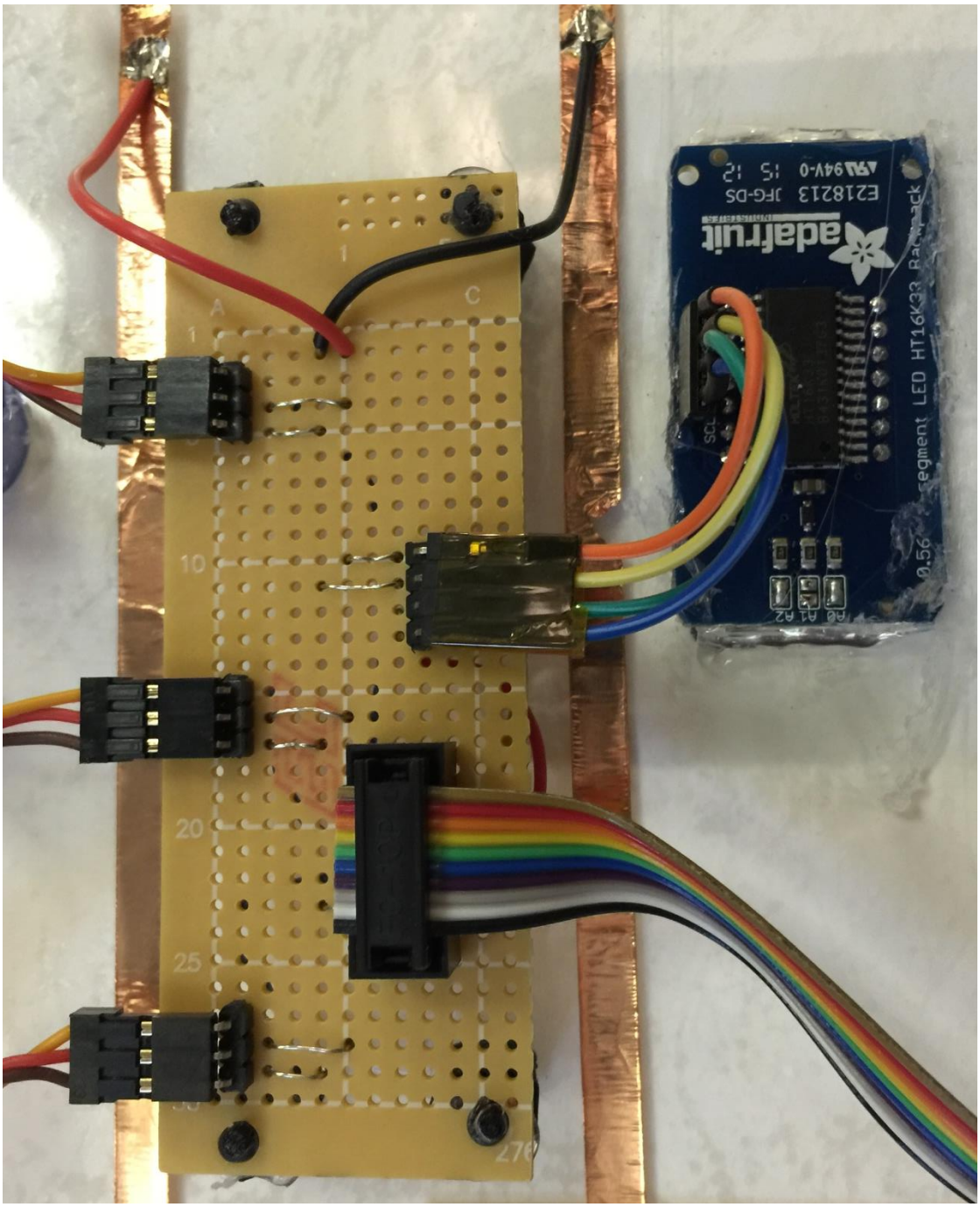
PWM		
ARDUINO	ATmega 1280	
PIN 13	PIN 26	PB7 (OC0A/OC1C/PCINT7)
PIN 12	PIN 25	PB6 (OC1B/PCINT6)
PIN 11	PIN 24	PB5 (OC1A/PCINT5)
PIN 10	PIN 23	PB4 (OC2A/PCINT4)
PIN 9	PIN 18	PH6 (OC2B)
PIN 8	PIN 17	PH5 (OC4C)

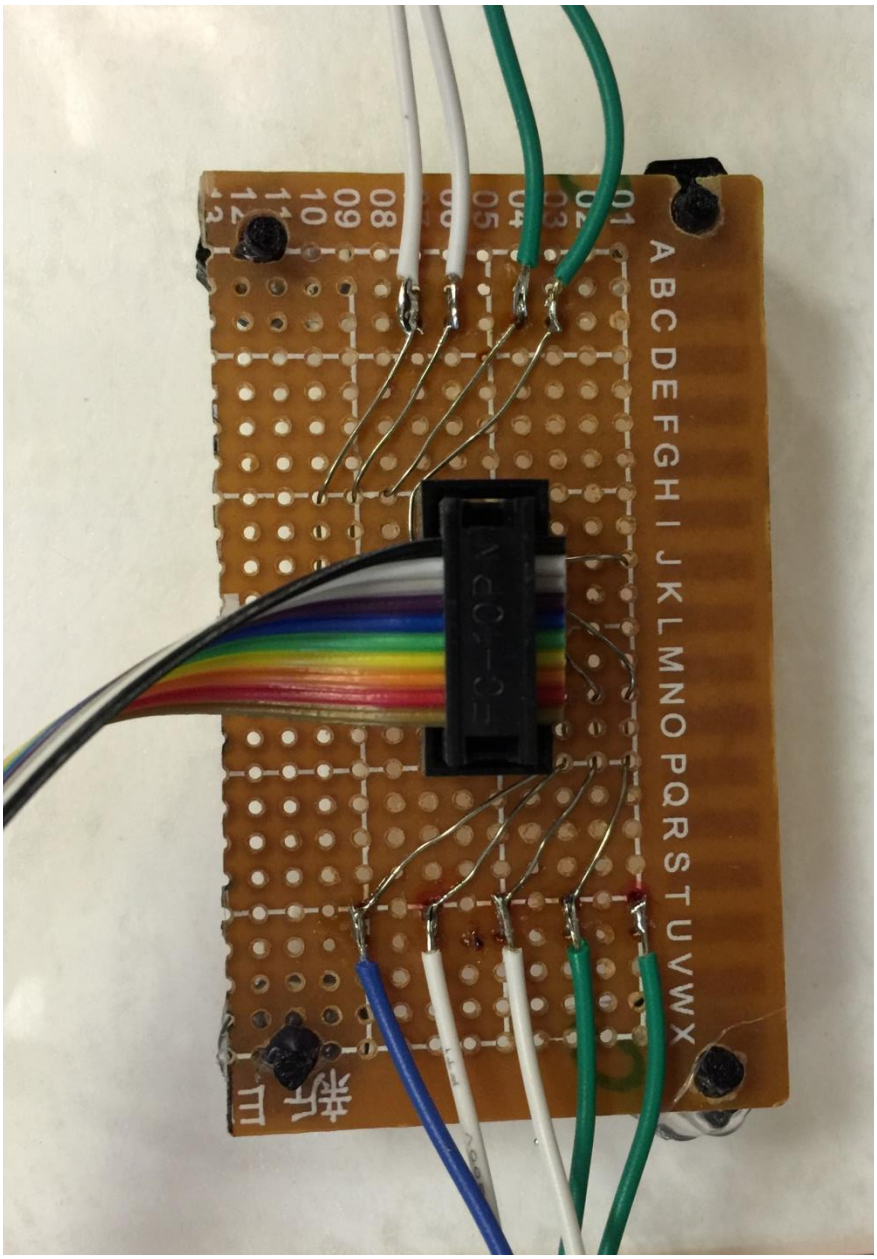
PIN 7	PIN 16	PH4 (OC4B)
PIN 6	PIN 15	PH3 (OC4A)
PIN 5	PIN 5	PE3 (OC3A/AIN1)
PIN 4	PIN 1	PG5 (OC0B)
PIN 3	PIN 7	PE5 (OC3C/INT5)
PIN 2	PIN 6	PE4 (OC3B/INT4)

COMMUNICATION		
ARDUINO	ATmega 1280	
PIN 1	PIN 3	PE1 (TXD0)
PIN 0	PIN 2	PE0 (RXD0/PCINT8)

PIN 14	PIN 64	PJ1 (TXD3/PCINT10)
PIN 15	PIN 63	PJ0 (RXD3/PCINT9)
PIN 16	PIN 13	PH1 (TXD2)
PIN 17	PIN 12	PH0 (RXD2)
PIN 18	PIN 46	PD3 (TXD1/INT3)
PIN 19	PIN 45	PD2 (RXD1/INT2)
PIN 20	PIN 44	PD1 (SDA/INT1)
PIN 21	PIN 43	PD0 (SCL/INT0)

DIGITAL		
ARDUINO	ATmega 1280	
PIN 22	PIN 78	PA0 (AD0)
PIN 23	PIN 77	PA1 (AD1)
PIN 24	PIN 76	PA2 (AD2)
PIN 25	PIN 75	PA3 (AD3)
PIN 26	PIN 74	PA4 (AD4)
PIN 27	PIN 73	PA5 (AD5)
PIN 28	PIN 72	PA6 (AD6)
PIN 29	PIN 71	PA7 (AD7)
PIN 30	PIN 60	PC7 (A15)
PIN 31	PIN 59	PC6 (A14)
PIN 32	PIN 58	PC5 (A13)
PIN 33	PIN 57	PC4 (A12)
PIN 34	PIN 56	PC3 (A11)
PIN 35	PIN 55	PC2 (A10)
PIN 36	PIN 54	PC1 (A9)
PIN 37	PIN 53	PC0 (A8)
PIN 38	PIN50	PD7 (T0)
PIN 39	PIN 70	PG2 (ALE)
PIN 40	PIN 52	PG1 (/RD)
PIN 41	PIN 51	PG0 (/WR)
PIN 42	PIN 42	PL7
PIN 43	PIN 41	PL6
PIN 44	PIN 40	PL5 (OC5C)
PIN 45	PIN 39	PL4 (OC5B)
PIN 46	PIN 38	PL3 (OC5A)
PIN 47	PIN 37	PL2 (T5)
PIN 48	PIN 36	PL1 (ICP5)
PIN 49	PIN 35	PL0 (ICP4)
PIN 50	PIN 22	PB3 (MISO/PCINT3)
PIN 51	PIN 21	PB2 (MOSI/PCINT2)
PIN 52	PIN 20	PB1 (SCK/PCINT1)
PIN 53	PIN 19	PB0 (/SS/PCINT0)


adafruit
A
C
1
10
20
25


A B C D E F G H I J K L M N O P Q R S T U V W X

Procedure on how to operate

Start MG Set No. 1 as follows:

NOTE: Perform the following steps to start MG Set #1 first, THEN repeat for MG Set #2.

a. Close motor circuit breaker.

b. Press AND hold "field flash" pushbutton.

NOTE: Generator voltage should rise above 220 VAC.

c. Release "field flash" pushbutton after voltage stabilizes.

NOTE: Maintain generator voltage above 220 VAC.

d. Rotate "Generator Voltage Adjust" potentiometer through full travel and record voltage adjustment range.

1. Record voltage adjustment range.

For Motor-Generator Set #1 _____VAC(min)
_____VAC(max)

For Motor-Generator Set #2 _____VAC(min)
_____VAC(max)

NOTE: Voltage should vary from approximately 220 to 300 VAC.

2. Record excitation current at min and max VAC.

For Motor-Generator Set #1 _____Amps(min)
_____Amps(max)

For Motor-Generator Set #2 _____Amps(min)
_____Amps(max)

e. Set "Generator Voltage Adjust" so voltmeter reads approximately 260 VAC.

1. Record excitation current.

For Motor-Generator Set #1 _____Amps
For Motor-Generator Set #2 _____Amps

f. Repeat steps to start MG Set No. 2.

a. Close generator circuit breaker (52-1).

Parallel Generators as follows:

a. Rotate "Generator No. 2 Synchronizer" switch to ON.

NOTE: Generator sets should parallel within 10 Sec.

b. Return "Generator No. 2 Synchronizer" switch to OFF.

c. Adjust "Generator No. 2 Voltage" so AC amps on both MG sets are approximately equal.

NOTE: AC amps will vary over time, this is normal. It may only be possible to keep current to within ±30 Amps of each other.

Increase voltage on Generator No. 1 to approximately 280 VAC AND check the following:

a. Generator No. 1 output breaker (52-1) trips after a time delay.

b. Generator No. 2 output breaker (52-2) does NOT trip.

c. Generator No. 2 output stabilizes at approximately 260 VAC.

d. Record Generator No. 2 excitation current. _____Amps

Reduce voltage on Generator No. 1 to approximately 260 VAC.

Parallel Generators as follows:

a. Rotate "Generator No. 1 Synchronizer" switch to ON.

NOTE: Generator sets should parallel within 10 Sec.

b. Return "Generator No. 1 Synchronizer" switch to OFF.

c. Adjust "Generator No. 1 Voltage" so AC amps on both MG sets are approximately equal.

Increase voltage on Generator No. 2 to approximately 280 VAC AND check the following:

a. Generator No. 2 output breaker (52-2) trips after a time delay.

b. Generator No. 1 output breaker (52-1) does NOT trip.

c. Generator No. 1 output stabilizes at approximately 260 VAC.

d. Record Generator No. 1 excitation current. _____Amps

Reduce voltage on Generator No. 2 to approximately 260 VAC.

Parallel Generators as follows:

a. Rotate "Generator No. 2 Synchronizer" switch to ON.

NOTE: Generator sets should parallel within 10 Sec.

b. Return "Generator No. 2 Synchronizer" switch to OFF.

c. Adjust "Generator No. 2 Voltage" so AC amps on both MG sets are approximately equal.

Decrease voltage on Generator No. 1 to approximately 240 VAC AND check the following:

NOTE: An increase in voltage setting on opposite generator may be necessary to trip under excited generator.

a. Generator No. 1 output breaker (52-1) trips after a time delay.

b. Generator No. 2 output breaker (52-2) does NOT trip.

Increase voltage on Generator No. 1 to approximately 260 VAC.

Parallel Generators as follows:

a. Rotate "Generator No. 1 Synchronizer" switch to ON.

NOTE: Generator sets should parallel within 10 Sec.

b. Return "Generator No. 1 Synchronizer" switch to OFF.

c. Adjust "Generator No. 1 Voltage" so AC amps on both MG sets are approximately equal.

Decrease voltage on Generator No. 2 to approximately 240 VAC AND check the following:

NOTE: An increase in voltage setting on opposite generator may be necessary to trip under excited generator.

a. Generator No. 2 output breaker (52-2) trips after a time delay.

b. Generator No. 1 output breaker (52-1) does NOT trip.

Open motor circuit breaker for MG Set No. 2.

Open Generator No. 1 output breaker.

Open motor circuit breaker for MG Set No. 1.

Arduino Mega program

This page defines some of the libraries used by the program to make the code lines / commands more easy to program.

```
/*

 Using the MP E-41.2, this program is to mimic a real function test.

 This is a working version, it works with procedure.

*/


#include <Servo.h>

#include <Wire.h>

#include "Adafruit_LEDBackpack.h"

#include "Adafruit_GFX.h"

Adafruit_7segment matrix = Adafruit_7segment();

#include "Arduino.h"
```

```
// the following sets the varibles and pin locations
int SyncStateCount;
int newpos;
Servo GenVolts1;
Servo ACAmps1;
Servo GenAmpsOut1;
int Gen1Close = LOW;
const int numReadings = 20;
int readings[numReadings];      // the readings from the analog input
int readIndex = 0;              // the index of the current reading
int total = 0;                  // the running total
int average = 0;
int value;
int pos = 50;
int pos1;
int cal;
int trip;
int once;
const int VoltKnob1 = A0;
int VoltSet1;
const int Exciter1 = 53;
int ExciterState1;
```

```
int ExciterCurrentState1;
int ExciterOn;
const int Sync1 = 11;
int SyncState1;
int ToggleState1;
int Repeat = HIGH;
int RanOnce;
const int Mg1Bkr = 42;
const int Mg1BkrT = 40;
int Mg1BkrState;
int Mg1BkrStateT;
int Mg1BkrCurrentState;
int Mg1RUNState;
int Mg1STOPState;
const int Gen1Bkr = 47;
const int Gen1BkrT = 45;
int Gen1BkrState;
int Gen1BkrStateT;
int Gen1BkrCurrentState;
int Gen1OPENState;
int Gen1CLOSEState;
const int MgRed1 = 38;
```

```
const int MgGrn1 = 36;

const int GenRed1 = 51;

const int GenGrn1 = 49;

Servo GenVolts2;

Servo ACAmps2;

Servo GenAmpsOut2;


// all of this stuff used to help stablize analog input

const int numReadings2 = 20;

int readings2 [numReadings2];      // the readings from the analog input

int readIndex2 = 0;              // the index of the current reading

int total2 = 0;                  // the running total

int average2 = 0;

int value2;

int pos2 = 50;

int pos21;

int cal2;

int trip2;

int once2;
```

```
// start naming all the devices
const int VoltKnob2 = A1;
int VoltSet2;
const int Exciter2 = 43;
int ExciterState2;
int ExciterCurrentState2;
int ExciterOn2;
const int Sync2 = 10;
int SyncState2;
int ToggleState2;
int Repeat2 = HIGH;
int RanOnce2;
const int Mg2Bkr = 22;
const int Mg2BkrT = 24;
int Mg2BkrState;
int Mg2BkrStateT;
int Mg2BkrCurrentState;
int Mg2RUNState;
int Mg2STOPState;
const int Gen2Bkr = 37;
const int Gen2BkrT = 35;
int Gen2BkrState;
```

```
int Gen2BkrStateT;
int Gen2BkrCurrentState;
int Gen2OPENState;
int Gen2CLOSEState;
const int MgRed2 = 26;
const int MgGrn2 = 28;
const int GenRed2 = 41;
const int GenGrn2 = 39;



// ================== Main SETUP ==============================
void setup() {

  // setting the 7 segment display to OFF

  matrix.begin(0x75); // A0, A2 jumpered
  matrix.clear();
  matrix.setBrightness (0);
  matrix.writeDisplay();
```

```
matrix.begin(0x70);
matrix.clear();
matrix.setBrightness (0);
matrix.writeDisplay();

// switches and buttons

pinMode(Exciter1, INPUT_PULLUP);
pinMode(Sync1, INPUT_PULLUP);
pinMode(Mg1Bkr, INPUT_PULLUP);
pinMode(Mg1BkrT, INPUT_PULLUP);
pinMode(Gen1Bkr, INPUT_PULLUP);
pinMode(Gen1BkrT, INPUT_PULLUP);

pinMode(Exciter2, INPUT_PULLUP);
pinMode(Sync2, INPUT_PULLUP);
pinMode(Mg2Bkr, INPUT_PULLUP);
pinMode(Gen2Bkr, INPUT_PULLUP);
pinMode(Mg2BkrT, INPUT_PULLUP);
pinMode(Gen2BkrT, INPUT_PULLUP);
```

```
// leds
pinMode(MgRed1, OUTPUT);
pinMode(MgGrn1, OUTPUT);
pinMode(GenRed1, OUTPUT);
pinMode(GenGrn1, OUTPUT);
pinMode(MgRed2, OUTPUT);
pinMode(MgGrn2, OUTPUT);
pinMode(GenRed2, OUTPUT);
pinMode(GenGrn2, OUTPUT);

GenVolts1.attach(5);
ACAmps1.attach(6);
GenAmpsOut1.attach(7);

GenVolts2.attach(2);
ACAmps2.attach(3);
GenAmpsOut2.attach(4);
```

```
  // set all the servos to move to indicate 0

  GenVolts1.write(180);
  ACAmps1.write(180);
  GenAmpsOut1.write(180);

  GenVolts2.write(180);
  ACAmps2.write(180);
  GenAmpsOut2.write(180);

  delay (15);

  // set all the OPEN/TRIP leds to ON

  digitalWrite(GenGrn1, HIGH);
  digitalWrite(MgGrn1, HIGH);

  digitalWrite(GenGrn2, HIGH);
  digitalWrite(MgGrn2, HIGH);
} // end void setup
```

```
// =============Main Loop and IF statements====================

void loop() {

 Mg1BkrState = digitalRead (Mg1Bkr); // read switch position LOW = ON, HIGH = OFF

  Mg1BkrStateT = digitalRead(Mg1BkrT);

  Gen1BkrState = digitalRead (Gen1Bkr);

  Gen1BkrStateT = digitalRead (Gen1BkrT);

  SyncState1 = digitalRead(Sync1); // will be used later to parallel both MGs.


  Mg2BkrState = digitalRead (Mg2Bkr); // read switch position LOW = ON, HIGH = OFF

  Mg2BkrStateT = digitalRead(Mg2BkrT);

  Gen2BkrState = digitalRead (Gen2Bkr);

  Gen2BkrStateT = digitalRead(Gen2BkrStateT);

  SyncState2 = digitalRead(Sync2); // will be used later to parallel both MGs.
```

```
if (Mg2BkrState == LOW || Mg2RUNState == HIGH) {
  Mg2Run();
}
if (Mg2BkrStateT == LOW && Mg2RUNState == HIGH) {
  Mg2RUNState = LOW;
  Mg2STOPState = HIGH;
  Mg2Stop();
}
if (Mg1BkrState == LOW || Mg1RUNState == HIGH) {
  Mg1Run();
}
if (Mg1BkrStateT == LOW  && Mg1RUNState == HIGH ) {
  Mg1RUNState = LOW;
  Mg1STOPState = HIGH;
  Mg1Stop();
}
if (Gen2BkrState == LOW && Mg2RUNState == HIGH) {
  Gen2OutBkrCls ();
}
if (Gen1BkrState == LOW  && Mg1RUNState == HIGH ) {
  Gen1OutBkrCls ();
}
```

```
  if (Gen1CLOSEState == HIGH && SyncState2 == LOW) {
    delay (9000);
    Gen2OutBkrCls ();
  }
  if (Gen2CLOSEState == HIGH && SyncState1 == LOW) {
    delay (9000);
    Gen1OutBkrCls ();
  }
  delay (10);
} // end Main Loop
```

```
void Mg1Run() {
  Mg1RUNState = HIGH;
  if (Mg1RUNState == HIGH ) {
    digitalWrite(MgGrn1, LOW);
    digitalWrite(MgRed1, HIGH);
    ExciterState1 = digitalRead(Exciter1);
    if (ExciterState1 == LOW || ExciterOn == HIGH ) {
      ExciterOn = HIGH;                    // helps keep in loop
      // this section to help stablize analog readings
      total = total - readings[readIndex];
      readings[readIndex] = analogRead(VoltKnob1);
      total = total + readings[readIndex];
      readIndex = readIndex + 1;
      if (readIndex >= numReadings) {
        readIndex = 0;
      }
      average = total / numReadings;
      // mapping to define servo movement that concides with Volts adjustment
      pos = map (average, 1023, 0, 1, 60);
      GenVolts1.write(pos);
      // mapping to define servo movement, part from POS/Volt meter
      pos1 = map (pos, 100, 30, 160, 122);
```

```
      ACAmps1.write(pos1);
      if ( Gen1CLOSEState == HIGH) {
        cal = map (pos, 100, 30, 10, 70);
        matrix.begin(0x75); // A0, A2 jumpered
        matrix.print(cal, DEC);
        matrix.setBrightness (2);
        matrix.writeDisplay();
        if (pos < 15 || pos > 50) {
          Gen1OutBkrOpn();
        }
      }
  } // end if EXCITER field
    RanOnce = LOW;
   } // end if MG1Run = HIGH
} // end Mg1Run
```

```
void Mg1Stop () {
  if (Mg1STOPState == HIGH ) {
    Mg1BkrCurrentState = Mg1BkrState;
    digitalWrite(MgRed1, LOW);
    digitalWrite(MgGrn1, HIGH);
    digitalWrite(GenRed1, LOW);
    digitalWrite(GenGrn1, HIGH);
    matrix.begin(0x75); // A0, A2 jumpered
    matrix.clear();
    matrix.setBrightness (0);
    matrix.writeDisplay();
    // set all the meters to zero
    for (pos1; pos1 < 180; pos1 += 1) {
      GenVolts1.write(pos1);
      ACAmps1.write(pos1);
      delay(15);
    }
    GenAmpsOut1.write(180);
    ExciterOn = LOW;
    RanOnce = HIGH;
  }
}
```

```
void Mg2Run() {
  Mg2RUNState = HIGH;
  if (Mg2RUNState == HIGH) {
    trip2 = LOW;
    digitalWrite(MgGrn2, LOW);
    digitalWrite(MgRed2, HIGH);
    ExciterState2 = digitalRead(Exciter2);
    if (ExciterState2 == LOW || ExciterOn2 == HIGH) {
      ExciterOn2 = HIGH;
      total2 = total2 - readings2[readIndex2];
      readings2[readIndex2] = analogRead(VoltKnob2);
      total2 = total2 + readings2[readIndex2];
      readIndex2 = readIndex2 + 1;
      if (readIndex2 >= numReadings2) {
        readIndex2 = 0;
      }
      average2 = total2 / numReadings2;
      // mapping to define servo movement that concides with Volts adjustment
      pos21 = map (average2, 1023, 0, 1, 60);
      GenVolts2.write(pos21);
      pos2 = map (pos21, 100, 30, 160, 122);
      ACAmps2.write(pos2);
```

```
    if (Gen2CLOSEState == HIGH) {
      cal2 = map (pos21, 100, 30, 10, 70);
      matrix.begin(0x70);
      matrix.print(cal2, DEC);
      matrix.setBrightness (2);
      matrix.writeDisplay();
      if (pos21 < 20 || pos21 > 50) {
        Gen2OutBkrOpn();
      }
    }
    RanOnce2 = LOW;
  }
 }
}
```

```
void Mg2Stop() {
  if (Mg2STOPState == HIGH) {
    Mg2BkrCurrentState = Mg2BkrState;
    digitalWrite(MgRed2, LOW);
    digitalWrite(MgGrn2, HIGH);
    digitalWrite(GenRed2, LOW);
    digitalWrite(GenGrn2, HIGH);
    matrix.begin(0x70);
    matrix.clear();
    matrix.setBrightness (0);
    matrix.writeDisplay();
    for (int pos21; pos21 < 180; pos21 += 1)
    {
      GenVolts2.write(pos21);
      ACAmps2.write(pos21);
      delay(15);
    }
    GenAmpsOut2.write(180);
    ExciterOn2 = LOW;
    RanOnce2 = HIGH;
  }
}
```

```
void Gen1OutBkrCls() {
  Gen1CLOSEState = HIGH;
  if (Gen1CLOSEState == HIGH) {
    digitalWrite(GenGrn1, LOW);
    digitalWrite(GenRed1, HIGH);
    if (Mg1RUNState == HIGH && ExciterOn == HIGH) {
      GenAmpsOut1.write(160);
      delay (100);
    }
    Gen1Close = HIGH;
  }
}
```

```
void Gen2OutBkrCls() {
  Gen2CLOSEState = HIGH;
  if (Gen2CLOSEState == HIGH) {
    digitalWrite(GenGrn2, LOW);
    digitalWrite(GenRed2, HIGH);
    if (Mg2RUNState == HIGH && ExciterOn2 == HIGH) {
      GenAmpsOut2.write(160);
      delay(100);
    }
  }
}
```

```
void Gen2OutBkrOpn() {
  Gen2CLOSEState = LOW;
  Gen2OPENState = HIGH;
  if (Gen2OPENState == HIGH) {
    trip2 = HIGH;                          // used as a condition for closing breaker
later.
    digitalWrite(GenRed2, LOW);
    digitalWrite(GenGrn2, HIGH);
    GenAmpsOut2.write(180);
    delay (15);
  }
}
```

```
void Gen1OutBkrOpn() {
  Gen1CLOSEState = LOW;
  Gen1OPENState = HIGH;
  if (Gen1OPENState == HIGH) {
    trip = HIGH;                          // used as a condition for closing breaker later.
    digitalWrite(GenRed1, LOW);
    digitalWrite(GenGrn1, HIGH);
    GenAmpsOut1.write(180);
    delay (15);
  }
}
```