```
/*remote portion of alexa app ATTIC / Greenhouse
 * 4/29/18
 * HCT11 / HC12 (configurable chan
 * compile with arduino nano (or equiv)
*/
#include <EEPROM.h>
// 2400 -117dbM 1000M
// 9600 -112dbm 600M default
int  Linkrate=9600,chanS=50,maskimage=1; // maskimage is local status of each configured bit
int  LinkMask= 0x1F,SETpin=10; //  1.1111 LinkMask defines which bits in rcv mak are configured in this remote
//**set these to match source**
int i,j,jj=0,k,m,n,cnt=1,Fd,S1off=0,S2off=0,FHoff=0,Pmpoff=0;
//int Dmap[8]={0,0,2,3,4,5,6,9}; // Dmapped pins (maps mask bit to pin)
String DSTR;
float T1,T2,H;
float arrayc[2];
const int F1=2,F2=3,vent=6,Cooler=9,Pump=5,Lite=4; //Mist1=7,Mist2=8,
int R[5],Bindex,interval=50,index,DHTinputPin=12,lcnt=1,icnt=1,Ht,Pt,Dt;
bool s1=0,alarm=0;
int sendM=0,profileis=0;
unsigned int defal[6]={50,9600,0,0,0,0};
// A1 assigned as fan format open-seperate/ 0-combined
//
void setup()
{
 Serial.begin(Linkrate);
 readEEPROM(); // read  & install EEPROM value
 for(i=2;i<9;i++)
 {
   digitalWrite(i,HIGH);
   pinMode(i,OUTPUT);
 }
 DSTR="AT+B"+String(Linkrate);
 sendHC();
 digitalWrite(Cooler,LOW);
 pinMode(Cooler, OUTPUT);
 digitalWrite(SETpin, HIGH); // HC12 set pin
 pinMode(SETpin, OUTPUT);
 digitalWrite(DHTinputPin, HIGH); // DHT11 pin
 pinMode(A3,INPUT_PULLUP);
 pinMode(11,INPUT_PULLUP);
 pinMode(A1,INPUT_PULLUP); //1=seperate F1 and F2 operation
}
//*************************************************
void loop()
{
 if (Serial.available())
 {
  Blueservice();
 }
 delay(10);
 lcnt--;
 icnt--;
 if (icnt<1)
 {
  digitalWrite(13,!digitalRead(13)); // flash led
  icnt=25;
 }
```

```
if (bitRead(maskimage,0)>0) // qualif enabled
{
  if(lcnt<1)
  {
    lcnt=300;

    s1=!s1;
    if(s1)
    {
      DHTinputPin=12;  // read attic temp @ S2
      rd_DHTsensor();
      T2=arrayc[0]+S2off   ;  // t1 alway 12 degr cooler!!
    }
    else
    {
      DHTinputPin=A0; // read outside temp/H @ S1
      rd_DHTsensor();
      T1=arrayc[0]+S1off;
      H=arrayc[1];
    }
/*
 greenhouse
  profiles are <70%,12deg delta, T2>74degr  pump at <60% : T2 inside temp
 profiles are <45%, 8deg delta, T2>69degr  pump at <40%
*/
/* attic*
  profiles are <70%,12deg delta, T2>85degr  pump at <60% : T2 inside temp
  profiles are <60%, 8deg delta, T2>72degr  pump at <50%
  In override F1 and F2 are enabled and Pump operates as before /w selected profile
*/
// (re)set profile parameters : default profile 0
    Fd=8;  // delta min T2-T1
    Ht=60; // max H
    Pt=50; // max for pump
    int ard=analogRead(A7);
/*    if(ard<100)
    {
      ard=512;
    } */
    Dt=72+(ard/1024)*8-4; // min T2 temp (attic/indoor temp)
    //
    if (bitRead(maskimage,1)>0) //profile parameters : default profile 1
    { // uses attic profile
      Fd=12; // delta
      Ht=70+FHoff; // max %
      Pt=60+Pmpoff; // max pump%
      Dt=85+(ard/1024)*8-4; // min T2 inside
    }
 //
      if (bitRead(maskimage,3)>0) // hi override
      {
        setFHI();       // F2 override only sets F2 (Hi)
        //qualPMP();
      }
      else
      {   // do high fan
       if((H<=Ht)&&(T1>=Dt)&&((T2-T1)>=Fd))  // qual Ht & T2(attic temp)
        {
```

```
            setFHI();
          }
          else // schedule F2 off
          {
           if (digitalRead(F2)<1)
 //else(qual for A1 and set vent/pumps off
           //qualPMP();
           //delay(50);
           //digitalWrite(Cooler,HIGH);    // led
           //delay(4500)
           {
            digitalWrite(F2,HIGH);
            delay(250);
           }
          }
        }
  // end T2 high
        if (bitRead(maskimage,2)>0) // low override
        {
          setFLow();       // F1 override only sets F1 (Low)
        }
        else    // do low fan
        {
         if (digitalRead(A1) || (!(digitalRead(A1)) && (digitalRead(F2)>0)))
         //if (digitalRead(F2)>0)
         {
           if((H<=Ht)&&(T1>=Dt))
           {
            setFLow();
           }
           else
           {
            digitalWrite(F1,HIGH);
            qualPMP(); //mpdigitalWrite(Pump,HIGH);
            delay(500);             // wait .5 seconds
           }
         }
         qualVent(); // close vent
        }
   }
   else //  loop timing failed
   {
     {}
   }
}
else // not enabled
{
    if (digitalRead(F2)<1)
    {
     digitalWrite(F2,HIGH);
     delay(500);
    }
    digitalWrite(Pump,HIGH);
    delay(150);
    digitalWrite(F1,HIGH);
    digitalWrite(Cooler,LOW);    // led
  delay(2000);                // wait 2 seconds
    digitalWrite(vent,HIGH);
```

```
}
// discrete switch controls
/*
 these are allocated on outputs with Linkmask bit set for these commands, on this project but are not used
 Mister1 could be renamed to Pump override and qualified in code
 Grow    could be renamed to Porch Light and used to enable a related exterior light
 these 3 pins could also simple be used to return status to the WeMo base of processed
 outputs or values such as power/battery status, F2 or pump output status. The code
 could also create any 'more specific' status and return it to the base for display,
 such as alarm status indicating smoke or motion, taken from an additional sensor.

 The Mister and Grow light controls could be applicable to an third controller, located somewhere
 else, the linkmask bits in this app set to zero and the Linkmask bits in that app set accoringly.

 if set in both locations, code should be corrected in 0ne of them to disable the status response
 returned in either the bit set/clear command or report request command.

*/
}
// ****************************

void qualVent() // qual vent off
{
  if(digitalRead(F1) & digitalRead(F2))
   {
    digitalWrite(vent,HIGH); //qualVent()
   }
}
// ****************************

 void qualPMP()
 {
    if((H<Pt) & (!digitalRead(F1) | !digitalRead(F2))) // qual max Pt-PUMP operates in override as before
    {
      digitalWrite(Pump,LOW);
    }
    else
    {
     digitalWrite(Pump,HIGH);
    }
 }
// ****************************

void setFLow()
{
   if(digitalRead(F1)>0)
   { //off
      if (digitalRead(vent))
      {
       digitalWrite(vent,LOW);
   delay(8000);          // wait 8 seconds
      }
      digitalWrite(F1,LOW);
      digitalWrite(Cooler,HIGH); // high true LED
   }
   qualPMP();
}
// ****************************
```

```
void setFHI()
{
  if (digitalRead(F2)>0)
  { //off
    if (!digitalRead(A1) && digitalRead(F1)<1)
    {
      digitalWrite(F1,HIGH);
      delay(50);
    }
    if (digitalRead(vent))
    {
      digitalWrite(vent,LOW);
  delay(8000);          // wait 8 seconds
    }
    digitalWrite(F2,LOW);
    digitalWrite(Cooler,HIGH); // high true LED
    delay(500);  // wait 1/2 seconds
  }
  qualPMP();
}
// ****************************

void Blueservice()  // serial link services
{
 bool d1=0;
 DSTR= char(Serial.read());
 DSTR+=char(Serial.read());
 DSTR+=char(Serial.read());
 if (DSTR=="E50") //format is E, chan, m, dat, cksum
 {
   // DSTR= char(Serial.read()); //skip '/'
    delay(35);  // read up to 15 characters(9600)
    if(Serial.available())
    {
      byte ci=Serial.parseInt();  //this is 0x40,0x41 or 0,1.. header
      byte cj=digitalRead(11)+2*digitalRead(A3);
      if(((ci & 1)==digitalRead(11)) || (digitalRead(A3)==0))  // chanindex(n) 1 or 0
      {
        k =Serial.parseInt();  //bit index ..mask,task specific bit
        m =Serial.parseInt();  //dat
        n =Serial.parseInt();  //cksum   based on: ci+k+m

// eg;   E5064,1,0,254    E50,0,1,1,253
//          Serial.println(String(ci)+" "+String(k)+" "+String(m)+" "+String(n));
        if(n==((k+m+ci)&0xFF)) // test cksum headr,mask+dat
        {
          Serial.println("er"); //error ..flush and return
        }
        else
        {  // checksum was OK
          for(i=0;i<7;i++)
          {
            if(bitRead(k,i)>0)
            {
              j=i; // convert to binary index
              i=8;
            }
```

```
          }
          if(LinkMask & k)
          {                 // a configured bit
// ***********************************************************************
            if((ci & 0x40)<1) // !report only
            {               // request is done on this remote
              if(j==1) // if profile setting
              {
                profileis=int(m);
                if (profileis>0)
                {
                  bitWrite(maskimage,j,HIGH);
                }
                else
                {
                  bitWrite(maskimage,j,LOW);
                }
              }
              else if (m==0x5E)
              {
                d1=1;
                bitWrite(maskimage,j,LOW);
                lcnt=1;
              }
              else if (int(m)==0xE5)
              {
                  bitWrite(maskimage,j,HIGH);
                  lcnt=1;
              }
             if(d1==1)
             {
              sendM=0xC5;
             }
//    response to source 1-C5,11000101,0-5C,01011100,flash-59,01011001
            }
            else
            {   // is report request
                sendM=0x5C;
                switch (j)
                {
                case 0: // on/off / flash if alarm bit set
                  if(bitRead(maskimage,j))
                  {
                   sendM=0xC5;
                   if(T2>100) // alarm attic > 100degrF
                   {
                     sendM=0x59;
                   }
                  }
                  break;
                case 1: // profile 1 or 2 : this is now passed as a number
                  if(bitRead(maskimage,j))
                  {
                   sendM=0xC5;
                  }
                  break;
                 case 2: //F1 on off or flash if over-ride
                  if(digitalRead(F1))
```

```
                {
                 sendM=0xC5;
                 if(bitRead(maskimage,2)){sendM=0x59;}
                }
                break;
              case 3:  //ovrd F2 on off or flash if over-ride
                if(digitalRead(F2))
                {
                 sendM=0xC5;
                 if(bitRead(maskimage,3)){sendM=0x59;}
                }
                break;
              case 4: // generic Switch./Light
                sendM=0xC5;
                if(digitalRead(Lite)){sendM=0x5C;}
                break;
               } // end switch
            }  // end report and service
            DSTR="";
            DSTR="R50,"+String(cj)+","+String(sendM)+","+String(sendM ^ 0xFF);
            Serial.println(DSTR);

          } // not configured
        }   // end checksum test
      }     // end bits match test
    } // no data
}  // emnd E50 test
else
{
  if(DSTR=="set")
  {
   d1=0;
   while(d1==0)
   {
    Serial.print("1$SetChan ");
    Serial.println(String(chanS));
    Serial.print("2$setBitRate ");
    Serial.println(String(Linkrate));
    Serial.print("3$S1$S2Offsets ");
    Serial.println(String(S1off)+" "+ String(S2off));
    Serial.print("4$Fan%Off$Pmp%Offset ");
    Serial.println(String(FHoff)+" "+ String(Pmpoff));
    flushthis();
    delay(1);
    j = 120;    // give yourself 30 seconds to answer
    while (!(Serial.available()) && (j>0))
    {
     delay(250);
     j--;
    }
    if ((j > 0) && (Serial.peek()!='x')) // found entry
    {

     j = Serial.parseInt();  // j is index
     switch (j) // main
     {
     case (1):
      chanS=Serial.parseInt();
```

```
          DSTR = String(chanS)+ "$";
          set_chan();
          Wrt_EEPROM(0, 8);
          break;
        case (2):

          Linkrate=Serial.parseInt();   // this is the link rate
          DSTR = String(Linkrate)+ "$";
          Wrt_EEPROM(8, 8);
          chgB();
//          Serial.begin(Linkrate); // after this instruction
//                the serial link rate and monitor must match
          delay(5);
          d1=1;
          break;
        case (3):
          S1off=Serial.parseInt();
          DSTR = String(S1off)+ "$";
          Wrt_EEPROM(16, 8);
//
          S2off=Serial.parseInt();
          DSTR = String(S2off)+ "$";
          Wrt_EEPROM(24, 8);
          //d1=1;
          break;

        case (4):
          FHoff=Serial.parseInt();
          DSTR = String(FHoff)+ "$";
          Wrt_EEPROM(32, 8);
//
          Pmpoff=Serial.parseInt();
          DSTR = String(Pmpoff)+ "$";
          Wrt_EEPROM(40, 8);
          //d1=1;
          break;
         } // end main switch
        }
        else
        {
          d1=1;  //  timeout or 'x'
          flushthis();
          Serial.println("et");
        }
      } // end while d1
    }
    else
    {
      d1=1;  //  any other entry
      flushthis();
    }
  }
  delay(35);
}

void Wrt_EEPROM(int adr, int j)
{
   k = DSTR.length();
```

```
  for (m = 0; m < j; m++)
  {
   EEPROM[adr + m] = 0; // clear data
   delay(100);
  }
  for (m = 0; m <  k; m++)
  {
   EEPROM[adr + m] = DSTR[m];  //        iisue with eewrt numbers  vrs characters
   delay(100);
  }
}

void flushthis()
{
   while(Serial.available()){Serial.read();} // flush
}

//
void set_chan() // pass channel number in i
{
    if (chanS<10)
    {
        DSTR="AT+C00"+String(chanS);
    }
    else if (i<100)
    {
       DSTR="AT+C0"+String(chanS);
    }
    else
    {
      DSTR="AT+C"+String(chanS);
    }
    sendHC();
}
void chgB()
{
  DSTR="AT+CIOBAUD"+String(Linkrate);
  sendHC();
}
void  sendHC()
  {
    digitalWrite(SETpin,LOW);
    delay(40);
    Serial.print(DSTR);
    digitalWrite(SETpin,HIGH);
    Serial.println();
    delay(80);
  }

void rd_DHTsensor()
{
 // act_chan is set and DHRinputPin defined by caller
  int Minterval = 0;
  unsigned long previousMicro = 0;
  unsigned long currentMicro  = 0 ;
  pinMode(DHTinputPin, OUTPUT);   // assert the sensor pin
  digitalWrite(DHTinputPin, LOW);
  Bindex = 5;  //count number of bytes to read
```

```
delay(17); // asser low for ~17mS
pinMode(DHTinputPin, INPUT);    // float pin
//                  seek pin low with timeout
index = 50;
delayMicroseconds(10); // float for ~10uS
i = 0;

while (index > 0)
{
 delayMicroseconds(2);
 bool DHTinputState = digitalRead(DHTinputPin);
 if (DHTinputState == LOW)          // init ack
 {
  index = 50;
  i = 0;
  while (index > 0)
  {
   delayMicroseconds(1);
   DHTinputState = digitalRead(DHTinputPin);
   if (DHTinputState == HIGH)       // sync gone high
   {
    index = 70;
    while (index > 0)
    {

     delayMicroseconds(2);
     DHTinputState = digitalRead(DHTinputPin);
     if (DHTinputState == LOW) // sync -> start predata low
     {
      while (Bindex > 0)
      {
       int bits = 8;   // number of bits/byte
       int Dinput = 0;
       while (bits > 0)
       {

        //seek pin high with timeout
        index = 50;
        while (index > 0)
        {
         DHTinputState = digitalRead(DHTinputPin);
         if (DHTinputState == HIGH) //start actual data
         {
          // digitalWrite(ledPin,HIGH);
          previousMicro = micros();
          index = 100;

          while (index > 0)
          {
           delayMicroseconds(1);
           DHTinputState = digitalRead(DHTinputPin);
           if (DHTinputState == LOW)
           {
            currentMicro = micros();
            Dinput = Dinput * 2;
            //    digitalWrite(ledPin,LOW);
            Minterval = currentMicro - previousMicro;
            // Serial.print(Minterval); // time measured uS
```

```
                    // Serial.println(" uS");

                    if (currentMicro - previousMicro >= interval)
                    {
                      Dinput++;
                    }
                    bits = bits - 1;
                    index = 0;
                  }
                  else
                  {
                    index--;
                  }
                 }
                }
                else
                {
                  index--;
                  delayMicroseconds(1);
                }
               }
              }
              R[(Bindex - 1)] = Dinput; //puts 1st byte in last loction eg: 4
              //  Serial.println(R[Bindex-1]);
              Bindex = Bindex - 1;
            }     // end Bindex>0
          }
          else
          {
            index--;
          }
         }         // end index > 0

       }
       else
       {
         index--;
       }
     }

     index--;
    }
    else
    {
     index--;
    }
  }

//if index==0 then error sensor not reponsive OR no sync pulse high
//                  calculate checksum
  i = R[4] + R[3] + R[2] + R[1];
  float x = 0;
  float f = 0;
//     qual  chksum
  if (i != R[0]) // deal with bit 9
  {
    Serial.println("Err");
  }
```

```
// ************************************
  else
  {                        // valid HCT data
   arrayc[0] = ((9*(R[2]+ (float(R[1])/10)))/5)+32; // HCT11 degrF
   //                                 //32-106degr,+-3.6deg
   arrayc[1] = R[4]+float((R[3])/10);
   if ((arrayc[1] < 20))
   {
      arrayc[1] = 20; // H is mininimu=20 %, 5%
   }
  }
}

void readEEPROM() // read and substitute default
{
 //             read EEPROM, set setting[] includes $ @end
 unsigned int setting[6];
 for (i=0;i<6;i++) // copy parameters in EEPROM to working array mister 8 bytes
 {
  k = 8; //read up to 8 charaters
  DSTR = "";
  j = 0;
  while ((EEPROM.read((i * 8) + j) != '$') && k > 0)
  {
   DSTR+=char(EEPROM.read((i * 8) + j));
   j++;
   k--;
   delay(40);
  }
  if (k > 0)             // if (k==-1) to force defaults
  {
    setting[i] = DSTR.toInt();
  }
  else
  {
     setting[i]=defal[i];
     DSTR = String(setting[i]) + "$";
     Wrt_EEPROM(i*8,8);
  }
 }
 chanS=setting[0];
 set_chan();
 Linkrate=setting[1];
 S1off=setting[2];
 S2off=setting[3];
 FHoff=setting[2];
 Pmpoff=setting[3];
 Serial.println("Initialzing");
}
```