program listing

//_240828_Solar_Switched_Sockets.ino

```
// PURPOSE & setup:

// Switching several sockets "on" while the solar panel is delivering power
// The boiler socket is hard-wired to the Arduino microprocessor
// For Remote Controlled (RC)sockets, a 433MHz transmitter-shield is used
// For the setting of the RC sockets, a power use of 500 W is assumed

// The solar input is measured with a small solar panel next to the main panels
// The small panel voltage (pV) is monitored via Arduino's analogue input A0
//  pV 1V equals  500 W input of main panels
//  pV 2V equals 1000 W
//  pV 2.5V equals 1400 W
//  pV 3.7 V equals 2050 W
// The boiler remains regulated by it's own thermostatic switch

// A 3-way switch allows for different CONDITIONS and restrictions
// 3-WAY SWITCH UP: BOILER socket "on" independent on value pV
//      a RC socket is switched "on", when there's sufficient power
// 3-WAY SWITCH DOWN: BOILER socket "off"
//      The remote sockets are switched depending on value pV
// 3-WAY SWITCH MIDDLE position: BOILER "pV regulated"
//      a RC socket is switched "on", when there's sufficient power

// Declaration of the used pins on the Arduino
const int pinMeasure = A0; // monitors the panelVoltage (pV)
const int pinPowerToBoiler = 5; // output to relay (pV regulated)
const int pinBoilerAlwaysOff = 10; // input, switches output 5 off when HIGH
const int pinBoilerAlwaysOn = 11; // input, switches output 5 on when HIGH
const int pinRCdata = 12; // Data to the 433 MHz transmitter

// Declaration and on-off settings (depending on pV small solar panel)
float volt_Boiler_Off = 2.5;  // 1400 W
float volt_Boiler_On  = 2.75; // 1500 W
float volt_RCA_Off = 1.1;    //  600 W
float volt_RCA_On  = 1.5;    //  800 W
float volt_RCB_Off = 2.0;    // 1100 W
float volt_RCB_On  = 2.45;   // 1350 W
float volt_RCC_Off = 2.9;    // 1600 W
float volt_RCC_On  = 3.2;    // 1800 W
float volt_BoilerPlusRC_Off = 3.45; // 2050 W
float volt_BoilerPlusRC_On  = 3.8;  // 2150 W
float pV = 0.0; // measured voltage small panel - MAX 5V!
long switchOffDelay = 20000; // millisecs delay before switching off

// Declarations and conditions
bool boiler_allowedOn = false; // regulated by pV
bool socketA_allowedOn = false;     // or switch
bool socketB_allowedOn = false;
bool socketC_allowedOn = false;
bool boiler_switchedOn = false; // runtime situation
bool socketA_switchedOn = false;
bool socketB_switchedOn = false;
bool socketC_switchedOn = false;

// Setting of the used RC switches
// delay times (pinData HIGH or LOW situations) in  µ seconds:
#define Bit24HighUp 998   //44 samplepoints
#define Bit24HighDown 476 //21 samplepoints
#define Bit24LowUp 295    //13 samplepoints
#define Bit24LowDown 1179 //52 samplepoints

#define Bit32HighUp 1429   //66 samplepoints
#define Bit32HighDown 590  //26 samplepoints
#define Bit32LowUp 408     //18 samplepoints
#define Bit32LowDown 1610  //71 samplepoints

#define StartDelay24LowUp 400
#define StartDelay24LowDown 2250
#define StartDelay32LowUp 408
#define StartDelay32LowDown 7188

// commands for the 3 sockets; 24 bit pattern is send 6x, 32 bit pattern 4x
bool S24[6][24] = {{0,0,1,1,0,0,0,0,1,0,1,1,1,0,1,0,0,0,1,1,0,1,0,1}, // A_ON
                   {0,0,1,1,0,1,1,1,0,0,0,0,1,1,0,1,0,1,0,0,0,1,0,1}, // A_OFF
                   {0,0,1,1,1,0,1,1,1,1,0,0,0,0,1,1,0,1,1,0,1,1,0,0}, // B_ON
                   {0,0,1,1,1,1,1,0,0,1,1,1,0,1,1,0,0,1,1,1,1,1,0,0}, // B_OFF
                   {0,0,1,1,0,1,1,0,1,1,1,0,0,0,0,1,1,0,1,1,1,1,1,0}, // C_ON
                   {0,0,1,1,0,1,0,1,0,1,1,0,1,0,0,0,0,0,0,1,1,1,1,0}};// C_OFF

bool S32[6][32] =
  {{0,0,1,1,1,0,1,0,1,1,1,1,0,1,1,1,1,1,1,1,1,0,0,1,1,0,1,0,1,0,1}, // A_ON
   {1,0,1,1,1,1,0,0,1,1,1,0,1,1,1,0,0,1,1,0,1,1,1,1,1,0,0,1,0,1,0,1}, // A_OFF
   {1,1,1,1,0,0,0,1,0,1,0,1,1,1,1,1,1,1,1,1,1,0,1,0,1,1,0,1,0,0,1,1}, // B_ON
   {1,0,0,0,1,0,1,1,1,0,0,0,1,1,1,0,1,0,0,1,1,0,0,0,1,0,0,1,0,0,1,1}, // B_OFF
   {1,1,1,1,1,1,1,1,1,1,0,1,1,0,0,1,1,0,1,0,1,0,0,1,1,1,0,1,1,0,1,1}, // C_ON
   {1,1,0,1,0,0,1,1,0,0,1,0,0,0,0,1,1,0,0,0,0,0,0,1,1,0,0,1,1,0,1,1}};// C_OFF

void setup()
{
  pinMode(pinPowerToBoiler, OUTPUT); // to relay on the socket
  pinMode(pinBoilerAlwaysOff, INPUT); // from 3-way switch
  pinMode(pinBoilerAlwaysOn, INPUT); // from 3-way switch
  pinMode(pinRCdata,OUTPUT);
  Serial.begin(9600);
  switchRC(1);  // switch the RC sockets off at start or reset
  switchRC(3);
  switchRC(5);
}
```

```
void loop()
{
  // measure small panel voltage pV
  pV = 5.0 * analogRead(pinMeasure) / 1024.0;
  // * serial output for diagnostics *
  Serial.println();
  Serial.print("     pV = ");
  Serial.print(pV);
  Serial.println();
  Serial.print(" pinBoilerAlwaysOn = ");
  Serial.print(digitalRead(pinBoilerAlwaysOn));
  Serial.println();
  Serial.print(" pinBoilerAlwaysOff = ");
  Serial.print(digitalRead(pinBoilerAlwaysOff));
  // * * 3-WAY SWITCH UP - BOILER SOCKET ON  * *
  // (switch puts 5V on pinBoilerAlwaysOn)
  if(digitalRead(pinBoilerAlwaysOn) == HIGH)
  {
    boiler_allowedOn = true;
    digitalWrite(pinPowerToBoiler, HIGH);
    boiler_switchedOn  = true;
    // one RC socket is allowed when boiler is on (MAX 500W assumed):
    switchRCsockets(boiler_switchedOn);
  }
  // * * * 3-WAY SWITCH DOWN - BOILER SOCKET OFF * * *
  // (switch puts 5V on pinBoilerAlwaysOff)
  if(digitalRead(pinBoilerAlwaysOff) == HIGH)
  {
    boiler_allowedOn = false;
    digitalWrite(pinPowerToBoiler, LOW);
    boiler_switchedOn  = false;
    // switch the RC sockets (while withBoiler == false)
    switchRCsockets(boiler_switchedOn);
  }
  // * * * 3-WAY SWITCH NEUTRAL - BOILER SOCKET pV REGULATED * * *
  if((digitalRead(pinBoilerAlwaysOn) == LOW) && (digitalRead(pinBoilerAlwaysOff) == LOW))
  {
    if(pV >= volt_Boiler_On)
    {
      boiler_allowedOn = true;
      if(boiler_switchedOn == false)
      {
        digitalWrite(pinPowerToBoiler, HIGH);
        boiler_switchedOn  = true;
      }
    }
    if((pV < volt_Boiler_Off) && (boiler_switchedOn == true))
    {
      delay(switchOffDelay); // delay and meassure again
      pV = 5.0 * analogRead(pinMeasure) /1024.0;
      if(pV < volt_Boiler_Off)
      {
        boiler_allowedOn = false;
        digitalWrite(pinPowerToBoiler, LOW);
        boiler_switchedOn  = false;
      }
    }
    switchRCsockets(boiler_switchedOn);
  }
  Serial.println();
  Serial.print(" boiler_switchedOn =  ");
  Serial.print(boiler_switchedOn);
  Serial.println();
  Serial.print(" socketA_switchedOn =  ");
  Serial.print(socketA_switchedOn);
  Serial.println();
  Serial.print(" socketB_switchedOn =  ");
  Serial.print(socketB_switchedOn);
  Serial.println();
  Serial.print(" socketC_switchedOn =  ");
  Serial.print(socketC_switchedOn);
  Serial.println();
  delay (4000);
}

void switchRCsockets(bool withBoiler)
{
  if (withBoiler == false) // switch the RC sockets while boiler off
  {
    if(pV >= volt_RCA_On)
    {
      socketA_allowedOn = true;
      if(socketA_switchedOn == false)
      {
        switchRC(0); // Socket A ON
        socketA_switchedOn = true;
      }
    }
    if((pV < volt_RCA_Off) && (socketA_switchedOn == true))
    {
      delay(switchOffDelay); // delay and meassure again
      pV = 5.0 * analogRead(pinMeasure) /1024.0;
      if(pV < volt_RCA_Off)
      {
        socketA_allowedOn = false;
        switchRC(1); // Socket A OFF
        socketA_switchedOn = false;
      }
    }
    if(pV >= volt_RCB_On)
    {
      socketB_allowedOn = true;
      if(socketB_switchedOn == false)
      {
```

```
   switchRC(2); // Socket B ON
   socketB_switchedOn = true;
  }
 }
 if((pV < volt_RCB_Off) && (socketB_switchedOn == true))
 {
  delay(switchOffDelay); // delay and meassure again
  pV = 5.0 * analogRead(pinMeasure) /1024.0;
  if(pV < volt_RCB_Off)
  {
   socketB_allowedOn = false;
   switchRC(3); // Socket B OFF
   socketB_switchedOn = false;
  }
 }
 if(pV >= volt_RCC_On)
 {
  socketC_allowedOn = true;
  if(socketC_switchedOn == false)
  {
   switchRC(4); // Socket C ON
   socketC_switchedOn = true;
  }
 }
 if((pV < volt_RCC_Off) && (socketC_switchedOn == true))
 {
  delay(switchOffDelay); // delay and meassure again
  pV = 5.0 * analogRead(pinMeasure) /1024.0;
  if(pV < volt_RCC_Off)
  {
   socketC_allowedOn = false;
   switchRC(5); // Socket C OFF
   socketC_switchedOn = false;
  }
 }
}

if(withBoiler == true) // switch the RC sockets while boiler on
{
 if(pV >= volt_BoilerPlusRC_On)
 {
  socketA_allowedOn = true;
  if(socketA_switchedOn == false)
  {
   switchRC(0); // Socket A ON
   socketA_switchedOn = true;
  }
 }
 if((pV < volt_BoilerPlusRC_Off) && (socketA_switchedOn == true))
 {
  delay(switchOffDelay); // delay and meassure again
  pV = 5.0 * analogRead(pinMeasure) /1024.0;
```

```
      if(pV < volt_BoilerPlusRC_Off)
      {
        socketA_allowedOn = false;
        switchRC(1); // Socket A OFF
        socketA_switchedOn = false;
      }
    }
  }
}

void switchRC(int commandNumber)
{             // switch commands depending on the used RC sockets
  int bitNumber;
  int bitPattern;
  // * * * * DIAGNOSTICS  * * * *
  Serial.println();
  Serial.print("Function switchRC called. Commandnumber: ");
  Serial.print(commandNumber);

  for (bitPattern = 0; bitPattern < 6; bitPattern++)  // 24 bit part 6 times
  {
    // start with extra LOW part
    digitalWrite(pinRCdata, HIGH);
    delayMicroseconds(StartDelay24LowUp);
    digitalWrite (pinRCdata, LOW);
    delayMicroseconds(StartDelay24LowDown);
    for (bitNumber = 0; bitNumber < 24; bitNumber++)
    {
      if (S24[commandNumber][bitNumber] == HIGH)
      {
        digitalWrite (pinRCdata, HIGH);
        delayMicroseconds(Bit24HighUp);
        digitalWrite (pinRCdata, LOW);
        delayMicroseconds(Bit24HighDown);
      }
      else
      {
        digitalWrite (pinRCdata, HIGH);
        delayMicroseconds(Bit24LowUp);
        digitalWrite (pinRCdata, LOW);
        delayMicroseconds(Bit24LowDown);
      }
    }
  }
  for (bitPattern = 0; bitPattern < 4; bitPattern++)  // 32bits part 4 times
  {
    //start with  an extra long LOW
    digitalWrite (pinRCdata, HIGH);
    delayMicroseconds(StartDelay32LowUp);
    digitalWrite (pinRCdata, LOW);
    delayMicroseconds(StartDelay32LowDown);
    for (bitNumber = 0; bitNumber < 32; bitNumber++)
  {
   if (S32[commandNumber][bitNumber] == HIGH)
   {
    digitalWrite (pinRCdata, HIGH);
    delayMicroseconds(Bit32HighUp);
    digitalWrite (pinRCdata, LOW);
    delayMicroseconds(Bit32HighDown);
   }
   else
   {
    digitalWrite (pinRCdata, HIGH);
    delayMicroseconds(Bit32LowUp);
    digitalWrite (pinRCdata, LOW);
    delayMicroseconds(Bit32LowDown);
   }
  }
 }
}

//Sketch uses 6000 bytes (18%) of program storage space. (Arduino Uno)
//Global variables use 736 bytes (35%) of dynamic memory
```