```
// THIS IS THE TEXT SOURcE CODE OF file: Auto_House.ino



/*
Arduino MICRO software to operate an automated model train home.

Written by: Bruce Kingsley
Date: July 05, 2014
Version: 1.0.0

Revsion Note:

  1.0.0   05-JUL-2014 RBK  First release.

*/

  void LightingEffects(void);
  void Effect(void);
  void WomanServoEffects(void);

#include <Servo.h>

#define NUM_OF_FADE_LIGHTS    7

#define LIGHT_TV         0
#define LIGHT_FRT_PORCH  1
#define LIGHT_LIVE_ROOM  2
#define LIGHT_SIDE_ROOM  3
#define LIGHT_KITCHEN    4
#define LIGHT_FRT_BEDRM  5
#define LIGHT_RER_BEDRM  6

#define EFFECT_DOOR_BELL_LIGHT_PIN    8
#define EFFECT_SOUND_PIN              12
#define EFFECT_WOMAN_SERVO_PIN        9

Servo WomanServoObject;          // create servo object to control a servo that rotates woman
in living room.
int WomanServoCommand;           // The last command sent to the woman servo (before mapping).
int WomanServoCommandPWMValue;   // The last value sent to the Servo Object.
unsigned long WomanServoNextOutput;  // The next time the servo output should be sent out.
#define SERVO_PWM_MIN  10        // The smallest PWM command we want to send to the servo.
#define SERVO_PWM_MAX  160       // The largest PWM command we want to send to the servo.
#define SERVO_VELOCITY 20        // The rate of rotation in milli-seconds. Bigger the number,
slower the rotation.

int FadeLightPorts[NUM_OF_FADE_LIGHTS];          // Store the digtal PWM ports the lights are
connect to.
bool FadeLightOnCommand[NUM_OF_FADE_LIGHTS];     // Is this light to be turned on or off.
bool FadeLightOn[NUM_OF_FADE_LIGHTS];            // Is this light on or off.
int LightFadeValue[NUM_OF_FADE_LIGHTS];          // The PWM value while in fading.
bool LightInFade[NUM_OF_FADE_LIGHTS];            // Light is currently fading.

unsigned long NextTelevisonFlicker; // The time the TV light level is changed.
#define TV_FLICKER_TIME 100         // How many milliseconds between TV flickers.

#define NUM_OF_SEQUENCES    15

int CurrentSequence;
unsigned long NextSequenceTime;
unsigned long SequenceDuration;  //Sequence in milliseconds

void setup()
{
  int i;
```

```
  // Setup the ports used for fade-able lights.

  FadeLightPorts[LIGHT_TV] = 13;  // The TV uses the PWM digital port 13.
  FadeLightPorts[LIGHT_FRT_PORCH] = 10;  // The front porch uses the PWM digital port 10.
  FadeLightPorts[LIGHT_LIVE_ROOM] = 11;  // The living room uses the PWM digital port 9.
  FadeLightPorts[LIGHT_SIDE_ROOM] = 6;  // The side rrom uses the PWM digital port 6.
  FadeLightPorts[LIGHT_KITCHEN] = 5;  // The kitchen uses the PWM digital port 5.
  FadeLightPorts[LIGHT_FRT_BEDRM] = 3;  // The front bedroom uses the PWM digital port 3.
  FadeLightPorts[LIGHT_RER_BEDRM] = 4;  // The rear bedroom uses the PWM digital port 4.



  for(i = 0; i!= NUM_OF_FADE_LIGHTS; i++)
    {
    pinMode(FadeLightPorts[i], OUTPUT);
    analogWrite(FadeLightPorts[i], 255);   //Hi value turns lights off.
    FadeLightOnCommand[i] = false;
    FadeLightOn[i] = false;
    LightFadeValue[i] = 255;
    LightInFade[i] = false;
    }

  //Setup the effect ports
  pinMode(EFFECT_DOOR_BELL_LIGHT_PIN, OUTPUT);
  pinMode(EFFECT_SOUND_PIN, OUTPUT);

   //Setup woman servo
  WomanServoCommand = 125;  // Start off with woman in the middle of rotation.
  WomanServoObject.attach(EFFECT_WOMAN_SERVO_PIN);  // Attaches the servo to the servo object.
  i = map(WomanServoCommand, 0, 255, SERVO_PWM_MIN, SERVO_PWM_MAX);  // scale it to use it with
the servo.
  WomanServoCommandPWMValue = i;
  WomanServoObject.write(i);
  WomanServoNextOutput = millis();

  NextTelevisonFlicker = millis();

  // Turn sound off
  digitalWrite(EFFECT_SOUND_PIN, HIGH);

  // Turn door bell button off
  digitalWrite(EFFECT_DOOR_BELL_LIGHT_PIN, HIGH);

  // Setup sequences
  CurrentSequence = 1;
  SequenceDuration = 1000;

  NextSequenceTime = millis();
  }

void loop()
{
  Effects();
  delay(1);

  if(millis() < NextSequenceTime) { return; } // If not time for next sequence, return from
loop.

  switch(CurrentSequence)
  {
    case 1:
      digitalWrite(EFFECT_DOOR_BELL_LIGHT_PIN, LOW);  // Turn on door bell button.
      digitalWrite(EFFECT_SOUND_PIN, HIGH);
      delay(50);
      digitalWrite(EFFECT_SOUND_PIN, LOW);
      FadeLightOnCommand[LIGHT_FRT_BEDRM] = true;
      SequenceDuration = 15288;
      break;
```

```
case 2:
  FadeLightOnCommand[LIGHT_RER_BEDRM] = true;
  SequenceDuration = 712;
  break;

case 3:
  FadeLightOnCommand[LIGHT_TV] = true;
  WomanServoCommand = 250;   // Face the TV
  SequenceDuration = 16167;
  break;

case 4:
  FadeLightOnCommand[LIGHT_KITCHEN] = true;
  FadeLightOnCommand[LIGHT_TV] = false;
  SequenceDuration = 1383;
  break;

case 5:
  FadeLightOnCommand[LIGHT_SIDE_ROOM] = true;
  SequenceDuration = 14451;
  break;

case 6:
  FadeLightOnCommand[LIGHT_LIVE_ROOM] = true;
  WomanServoCommand = 125;   // Face the Man
  SequenceDuration = 1380;
  break;

case 7:
  SequenceDuration = 10170;
  break;

case 8:
  FadeLightOnCommand[LIGHT_FRT_PORCH] = true;
  WomanServoCommand = 0;   // Face the Door
  SequenceDuration = 8940;
  break;

case 9:
  FadeLightOnCommand[LIGHT_FRT_PORCH] = false;
  SequenceDuration = 964;
  break;

case 10:      // Start of goodnight
  FadeLightOnCommand[LIGHT_SIDE_ROOM] = false;
  SequenceDuration = 2298;
  break;

case 11:
  FadeLightOnCommand[LIGHT_RER_BEDRM] = false;
  SequenceDuration = 2298;
  break;

case 12:
  FadeLightOnCommand[LIGHT_KITCHEN] = false;
  SequenceDuration = 2298;
  break;

case 13:
  FadeLightOnCommand[LIGHT_LIVE_ROOM] = false;
  SequenceDuration = 2298;
  break;

case 14:
  FadeLightOnCommand[LIGHT_FRT_BEDRM] = false;
  SequenceDuration = 15000;
  break;
```

```
    case 15:
      digitalWrite(EFFECT_DOOR_BELL_LIGHT_PIN, HIGH);  // Turn off door bell button.
      SequenceDuration = 5000;
     break;


  }

  //Setup for the next sequence
  NextSequenceTime = millis() + SequenceDuration;
  CurrentSequence++;
  if(CurrentSequence == NUM_OF_SEQUENCES + 1) { CurrentSequence = 1; } // If at the end, start
all over.

  return;
}

void Effects(void)
{
  // This function fades a light on or off
  int i;

  TelevisonEffect();  // If televison is on, create a flicker effect.
  WomanServoEffects();

  for(i = 0; i!= NUM_OF_FADE_LIGHTS; i++)
    {
    if(i != LIGHT_TV) // We handle the TV a different way.
      {
      if(FadeLightOn[i] != FadeLightOnCommand[i]) // The light command has changed.
         {
         FadeLightOn[i] = FadeLightOnCommand[i];
         LightInFade[i] = true;
         if(FadeLightOn[i] == true) { LightFadeValue[i] = 255; } else { LightFadeValue[i] =
0; }
         }

      if(LightInFade[i] == true) // We're in the transistion of turning a light on or off.
        {
        if(FadeLightOn[i] == true)
          {
          LightFadeValue[i]--;
          if(LightFadeValue[i] == 0) { LightInFade[i] = false; }
          }
          else
          {
          LightFadeValue[i]++;
          if(LightFadeValue[i] == 255) { LightInFade[i] = false; }
          }

        analogWrite(FadeLightPorts[i], LightFadeValue[i]);  // Write new PWM value.
        }
      }
    }

  return;
}


void TelevisonEffect(void)
{
  int TVLEDCommand;
  if(FadeLightOnCommand[LIGHT_TV] == false)
    {
    analogWrite(FadeLightPorts[LIGHT_TV], 255); // Remember, we need a hi vaue to turn a light
off.
    return;
```

```
    }

   if(millis() < NextTelevisonFlicker) { return; }  // If not ready to change TV brightness,
return.

   TVLEDCommand = random(0, 150);
   analogWrite(FadeLightPorts[LIGHT_TV], TVLEDCommand);
   NextTelevisonFlicker = millis() + TV_FLICKER_TIME;

   return;
}


void WomanServoEffects(void)
{

  int CommandMapped;

  CommandMapped = map(WomanServoCommand, 0, 255, SERVO_PWM_MIN, SERVO_PWM_MAX);

  if(CommandMapped != WomanServoCommandPWMValue)
    {
    if(millis() < WomanServoNextOutput) { return; }
    WomanServoNextOutput = millis() + SERVO_VELOCITY;

    if(CommandMapped > WomanServoCommandPWMValue)  // Do we need to go CW or CCW?
       {
       WomanServoCommandPWMValue++;
       }
       else
       {
       WomanServoCommandPWMValue--;
       }

    WomanServoObject.write(WomanServoCommandPWMValue); // Send incremental command to servo's
PWM.
    }

  return;
}
```