```
/*              Appendiks C-I: Arduino-softwarens kildekode                */
/***************************************************************************/
/*                            PROGRAM HEADER
** Files in sketch:
**   Master           * Contains definitions, basic and misc. functions
**                      for the main device
**   Communication    * Contains functions for communication with slaves
**                      and the PC
**   Games            * Contains functions for running the actual games
*/
/***************************************************************************/

/*                                MASTER                                   */

#define     BAUD     115200// Baud rate for all RS232-communication
#define      MAXINP      30// Max chars in input from PC
#define      MINBTN       1// Button-numbering minimum (digital IO pins)
#define      MAXBTN      24// Button-numbering maximum (digital IO pins)
#define TH 21

// Color definitions
#define OFF 13
#define RED 38
#define GREEN 63
#define BLUE 88
#define YELLOW 113
#define PURPLE 138
#define TURQUOISE 163
#define WHITE 188
#define PULSE 213
#define EXTRA 238

char     InpFromPc[MAXINP];// Used for reading input from the PC
int    PcStatus     =     0;// Goes 1 if PC-handshaking was successful
int  SwitchGameFlag   =  0;// For aborting a running game to switch to a new on
int     Game      =      -1;// Current game
int    BtnsPressed[MAXBTN];// Holds info for all pressed buttons
int      I       =       0;// Used for reading serial input from the PC

void setup() {
  int p;
  for (p = MINBTN+TH; p <= MAXBTN+1+TH; p++) {
    pinMode(p, INPUT);      // Set button as digital input
  }

  // Master
  Serial.begin(BAUD);
  // Request status from master
  Serial.print("STATUS\n" );
  // Slave 1
  Serial1.begin(BAUD);
  // Slave 2
```

```
  Serial2.begin(BAUD);
}

// Switches to a specific game
void SwitchGame(int gameNo) {
  // Set the SwitchGameFlag, breaking the game-loops
  SwitchGameFlag = 1;

  // Set the global Game variable to the game
  Game = gameNo;

  // Turn off all displays
  int a;
  for (a = 1; a <= 24; a++)
     SetDisplayColor(a, OFF);
}

// Fills the BtnsPressed-array with all the pressed buttons
// Clear the button input with ClearButtonInput()
void CollectButtonInput() {
  int a;
  for (a = MINBTN+TH; a <= MAXBTN+TH; a++) {
    if (digitalRead(a) == HIGH)
       BtnsPressed[a-TH] = 1;
  }
}

// Clears the button-input set with CollectButtonInput()
void ClearButtonInput() {
  int a;
  for (a = MINBTN; a <= MAXBTN; a++) {
     BtnsPressed[a] = 0;
  }
}

// Returns one of the random colors that are defined
// Should be extended
int RandomColor() {
  int color =random(1, 8); // Random no. 1-7
  switch(color) {
    case 1:
       color = RED;
      break;
    case 2:
       color = GREEN;
      break;
    case 3:
       color = BLUE;
      break;
    case 4:
       color = YELLOW;
      break;
```

```
    case 5:
       color = PURPLE;
      break;
    case 6:
       color = TURQUOISE;
      break;
    case 7:
       color = WHITE;
      break;
  }
  return  color;
}

int  RandomColorRGB()  {
  int  color  =random(1, 4);  // Random no. 1-3
  switch(color) {
    case 1:
       color = RED;
      break;
    case 2:
       color = GREEN;
      break;
    case 3:
       color = BLUE;
      break;
     }
  return  color;
}

void  loop() {
        CollectSerialInput();// Collect serial input from the computer, listeni
  switch      (Game)        {// Switch to the appropriate game
    case 0:
         SwitchGameFlag = 0;// Setting SwitchGameFlag causes the Game-loops to
       Game0();
      break;
    case 1:
       SwitchGameFlag = 0;
       Game1();
      break;
    case 2:
       SwitchGameFlag = 0;
       Game2();
      break;
    case 3:
       SwitchGameFlag = 0;
       Game3();
      break;
    case 4:
       SwitchGameFlag = 0;
       Game4();
      break;
```

```
    case 5:
       SwitchGameFlag = 0;
       Game5();
      break;
    case 6:
       SwitchGameFlag = 0;
       Game6();
      break;
  }
}
```

```
/*            Appendiks C-II: Arduino-softwarens kildekode              */
/*                          COMMUNICATION                                */

/***********************/
/*        RECEIVE       */
/***********************/
// Collects, assembles and processes any serial input from the PC
void  CollectSerialInput()  {
  // If there's something to read from the PC
  while (Serial.available()) {
    // The InpFromPc[]-array contains the partially-processed command
     InpFromPc[I] =Serial.read();
    // If there's a newline character, process the input
    if  (InpFromPc[I]  =='\n') {

      // Read the input...

      // If its an OK signal
      if  (InpFromPc[0]  =='O'  &&  InpFromPc[1]  =='K') {
         PcStatus = 1;
      }

      // If its a switch-game signal (G01 for game one)
      if  (InpFromPc[0]  =='G') {
        // Fetch the game no, -48 converts from ASCII decimal to int
        int  game = 10*(int(InpFromPc[1])-48);
        game += (int(InpFromPc[2])-48);

        // Switch to the game
         SwitchGame(game);

        // Let the PC know that we're switching to another game
        Serial.print("Switching to game " );
        Serial.print(game);
        Serial.print("\n");
      }

      // If its a switch-color-on-display-command (D01,255 for display 01 colo
      // This command is only for testing/debugging purposes
      if  (InpFromPc[0]  =='D') {
        // Fetch the display no
        int  disp = 10*(int(InpFromPc[1])-48);
         disp +=int(InpFromPc[2])-48;

        // Get the color no
        int  color = 100*(int(InpFromPc[4])-48);
        color += 10*(int(InpFromPc[5])-48);
         color +=int(InpFromPc[6])-48;

        // Let the PC know we're switching the color on a display
        Serial.print("Switching display " );
        Serial.print(disp);
```

```
        Serial.print(" to color ");
        Serial.print(color);
        Serial.print("\n");

        // Send the switch-color command
         SetDisplayColor(disp, color);
      }

      // Flush the string array
       InpFromPc[0] ='0'; // Numeric zero, not the letter
      I = -1;
    }
    I++;
  }
}

/**********************/
/*        TRANSMIT      */
/**********************/

// Sets a specific color on a specific display
void  SetDisplayColor(int  dispNo,int color) {
  // Don't allow a full pwm signal - won't work when using timers
  if (color == 255)
    color = 254;
  // Don't allow an empty pwm signal for the same reason
  if (color == 0)
    color = 1;

  // The "set display color"-command is formatted like this:
  //  DXX,YYY
  //  Where XX is the number of the display, 00-24
  //  And YYY is the color, 001-254
  if  (dispNo  <=  12)  {// Disp 1-12: Slave 1
    Serial1.print("D");
    if (dispNo < 10)
      Serial1.print("0");
    Serial1.print(dispNo);
    Serial1.print(",");
    if (color < 100)
      Serial1.print("0");
    if (color < 10)
      Serial1.print("0");
    Serial1.print(color);
    Serial1.print("\n");
  }else if (dispNo > 12) {// Disp 13-24: Slave 2
    dispNo -= 12;
    Serial2.print("D");
    if (dispNo < 10)
      Serial2.print("0");
    Serial2.print(dispNo);
    Serial2.print(",");
```

```
    if (color < 100)
      Serial2.print("0");
    if (color < 10)
      Serial2.print("0");
    Serial2.print(color);
    Serial2.print("\n");
  }
}

// Submits a score to the PC
void SubmitScore(long score) {
  Serial.print("SCORE:");
  Serial.print(score);
  Serial.print("\n");
}
```

```
/*            Appendiks C-III: Arduino-softwarens kildekode            */
/*                                GAMES                                */

/* Game #0
   Puts all displays in "Pulse"-mode. Used for showing off in a standby-phase
*/
void  Game0(void) {
  //  Variable  initialization
  int  state = 0;

  // The game will continue to run until a "switch game"-command is received
  while  (!SwitchGameFlag)  {
    // Collect commands from the PC
     CollectSerialInput();

    // Actions to do in the first state
    if (state == 0) {
      // Send the "pulse"-command to all displays
      int  i;
      for  (i = 1; i <= 24; i++)
         SetDisplayColor(i, PULSE);

      // Send the game to the next state
       state = 1;
    }

    // Here, state 1 could be initiated with:
    // else if (state == 1) { Code for state 1 }
  }
}

/* Game #1
   A random display lights up red, green or blue after 2-20 secs
   The time it takes (in ms) for the player to click the corresponding
   button is sent to the PC.
*/
void  Game1(void) {
  int  state = 0;
  long  startval;
  int  waitMs,  dispNo;
  while  (!SwitchGameFlag)  {
     CollectSerialInput();
    if  (state == 2)
       CollectButtonInput();

    if  (state == 0)  {
       dispNo =random(1,   25);// random gives a num between min and max-1
      waitMs = 1000;//random(2000,  20001);  // Wait  for  2-20  secs
       startval =millis();
      state = 1;
     }else if  (state == 1)  {
      if  (millis()  >=  startval+waitMs)  {
```

```
          SetDisplayColor(dispNo, RandomColor());
          startval =millis();
         state = 2;
          ClearButtonInput();
       }
      }else if (state == 2) {
       if (BtnsPressed[dispNo] == 1) {
          SubmitScore(millis()-startval);
          SetDisplayColor(dispNo, OFF);
         state = 0;
       }
     }
   }
}

/* Game #2
   A random color is picked (toClickColor). The corners light up in this color
   One display lights up with toClickColor, and numRandoms (starting as 0) ran
   non-corner displays light up with !toClickColor.
   The number of ms it takes the user to click the display with toClickColor
   sent to the PC.
   numRandoms is increased by 1 for each round, though maxes at 19.
*/
void Game2(void) {
  // Declare and init variables
  int state = 0;
  int numRandoms, toClickColor, dispNo = 0, i, randDispNo, isTaken;
  long  stTime;// Holds the starting time - used for calculating score

  int  disps[20];// Holds the displays that are already used
   disps[1] = 1;// Display 1, 6, 19 and 24 are used to show what color, the p
  disps[6] = 1;
  disps[19] = 1;
  disps[24] = 1;

  while (!SwitchGameFlag) {
      CollectSerialInput();// Make sure to collect serial input, in case we wa
    if (state == 2)// If we need button input (only needed in state 2), colle
       CollectButtonInput();

    if (state == 0) {// State 0: Show a random color (toClickColor) in the c
       numRandoms = 0;// The number of "confusion fields" that should be turne
       toClickColor = RandomColor();
        SetDisplayColor(1, toClickColor);
        SetDisplayColor(6, toClickColor);
        SetDisplayColor(19, toClickColor);
        SetDisplayColor(24, toClickColor);
        stTime =millis(); // Start the clock
      state = 1;
      dispNo = 0;
     }else if (state == 1) {
      if (millis() >= stTime+3000) {// When the clock has been ticking for 3
```

```
    // Find the random display, the player should click
    while (dispNo == 6 || dispNo == 19 || dispNo == 0) {// Must not be a
       dispNo =random(2,  24);// 2-23
    }
       SetDisplayColor(dispNo, toClickColor);// Light up the display
     disps[dispNo] = 1;// Make sure its not overwritten by a random color

    // The number of random displays should increase by one per turn
    for (i = 0; i < numRandoms; i++) {
      // Get a random display not already used
       randDispNo =random(1,  24);// Random no. between 1-23

      // Keep trying to find a random color till something not already use
      int isTaken = 1;
      while (isTaken == 1) {
        if (disps[randDispNo] == 1)
           randDispNo =random(1, 24);
        else
          isTaken = 0;
      }

      // Get a random color thats not toClickColor
      int randColor = RandomColor();
      while (randColor == toClickColor)
         randColor = RandomColor();

      // Set the random display to the random color
       disps[randDispNo] = 1;
        SetDisplayColor(randDispNo, randColor);
    }

    // Restart the clock
     stTime =millis();
    state = 2;
  }
}else if (state == 2) {
  if (BtnsPressed[dispNo] == 1) {// When the player clicked the right di
    // Save the score
     SubmitScore(millis()-stTime);

    // Turn off all displays, except the corners
    for (i = 2; i < 24; i++) {
      if (i != 6 && i != 19) {
         SetDisplayColor(i, OFF);
        disps[i] = 0;
      }
    }

    // If we haven't hit the max no. of random fields, give one more field
    if (numRandoms < 19)
       numRandoms++;
```

```
        // Restart the clock and go back to state 1
         stTime =millis();
        state = 1;
         ClearButtonInput();
        dispNo = 0;
      }
    }
  }
}

//  Vendespil
void  Game3(void) {
  int state = 0, score, disp1, disp2, numcorrects, i;
  int  colors[25],  guessed[25];
  while  (!SwitchGameFlag)  {
     CollectSerialInput();
    if (state == 1)
       CollectButtonInput();

    if (state == 0) {
       numcorrects = 0;
       score = 24;
      for (i = 0; i < 25; i++) {
         colors[i] = 0;
         guessed[i] = 0;
      }
       disp1 = disp2 = 0;
      for (i = 0; i < 12; i++) {
        int  theColor  =  RandomColor();
        while  (colors[disp1]  !=  0)
           disp1 =random(1, 25);
        while  (colors[disp2]  !=  0)
           disp2 =random(1, 25);
         colors[disp1] = colors[disp2] = theColor;
      }

       state = 1;
     }else if (state == 1) {
       disp1 = disp2 = 0;
      for (i = 1; i <= 24; i++) {
        if  (BtnsPressed[i]  ==HIGH  &&  !guessed[i])  {
           SetDisplayColor(i, colors[i]);
          if  (!disp1)
             disp1 = i;
          if (disp1 && !disp2 && disp1 != i)
             disp2 = i;

          if (disp1 && disp2) {
            if  (colors[disp1]  ==  colors[disp2])  {
               numcorrects++;
                guessed[disp1] = guessed[disp2] = 1;
```

```
              score++;
             }else {
              score--;
            }
          }
        }
        if (!guessed[i] && BtnsPressed[i] ==LOW) {
           SetDisplayColor(i, OFF);
        }
      }

      if (numcorrects >= 12)
         state = 2;

       ClearButtonInput();
     }else if (state == 2) {
       SubmitScore(score);
      state = 0;
    }
  }
}

//  Knaptest
void  Game4(void) {
  int state = 0;

  while (!SwitchGameFlag) {
     CollectSerialInput();
     CollectButtonInput();

    if (state == 0) {
      int i;
      for (i = 0; i <= MAXBTN; i++) {
        if (BtnsPressed[i] == 1) {
           SetDisplayColor(i, RED);
         }else {
           SetDisplayColor(i, OFF);
        }
      }
       ClearButtonInput();
    }
  }
}

void  Game5(void) {
  int state = 0;
  int i;
  long  stval;
  while (!SwitchGameFlag) {
     CollectSerialInput();

    if (state == 0) {
```

```
        stval =millis();
       state = 1;
      }else if (state == 1) {
       if (millis() >= stval+200) {
         int color = RandomColor();
         for (i = 0; i <= MAXBTN; i++)
             SetDisplayColor(i, color);
        state = 0;
       }
     }
   }
}

void Game6(void)
{
    int state = 0;
    int i, color;
    int LedsActive[3];
    int TimesWithSameColor;
    int TimesPlayedTotal = 0;
    long starttime;
    int dispNo[3];
    int olddispNo[3];
     dispNo[0] = dispNo[1] = dispNo[2] = dispNo[3] = 0;
     olddispNo[0] = olddispNo[1] = olddispNo[2] = olddispNo[3] = 0;
    while   (!SwitchGameFlag)
    {
       CollectSerialInput();

      if (state == 0)
      {
         TimesPlayedTotal++;
         starttime =millis();
         color = RandomColorRGB();
        state = 1;
         TimesWithSameColor = 0;
       }

      else if (state == 1)
      {
        if (millis() <= starttime+3000)
        {
          for (i = 0; i <= 24; i++)
           {
               SetDisplayColor(i, color);
          }
        }
        else { state = 2; }
      }

      else if (state == 2)
      {
```

```
    olddispNo[1] = dispNo[1];
    olddispNo[2] = dispNo[2];
    olddispNo[3] = dispNo[3];
  while((dispNo[1]  =  RandomColumn1And2())  ==  olddispNo[1]);
  while((dispNo[2]  =  RandomColumn3And4())  ==  olddispNo[2]);
  while((dispNo[3]  =  RandomColumn5And6())  ==  olddispNo[3]);

     SetDisplayColor(dispNo[1],  color);
    SetDisplayColor(dispNo[2],  color);
    SetDisplayColor(dispNo[3],  color);
   LedsActive[1] = LedsActive[2] = LedsActive[3] = 1;

    starttime =millis();
  for(i = 0;  i <= 24;  i++)
  {
    int  randomcolor;
    while((randomcolor  =  RandomColorRGB())  ==  color)  {  }
    if(i  ==  dispNo[1]  ||  i  ==  dispNo[2]  ||  i  ==  dispNo[3])  {
     }else  {
        SetDisplayColor(i,  randomcolor);
     }
  }
   state = 3;
}
else  if(state == 3)
{
    CollectButtonInput();
  if(BtnsPressed[dispNo[1]] == 1)
  {
      SetDisplayColor(dispNo[1],  OFF);
     LedsActive[1] = 0;
     ClearButtonInput();
  }
  else  if(BtnsPressed[dispNo[2]] == 1)
  {
      SetDisplayColor(dispNo[2],  OFF);
     LedsActive[2] = 0;
     ClearButtonInput();
  }
  else  if(BtnsPressed[dispNo[3]] == 1)
  {
      SetDisplayColor(dispNo[3],  OFF);
     LedsActive[3] = 0;
     ClearButtonInput();
  }

  if(LedsActive[1]  ==  0  &&  LedsActive[2]  ==  0  &&  LedsActive[3]  ==  0)
  {
     SubmitScore(millis()-starttime);
      TimesWithSameColor  =  TimesWithSameColor  +  1;
    if(TimesWithSameColor > 4) {
      state = 0;
```

```
            }else {
              state = 2;
             for (i = 0; i <= 24; i++)
             {
                 SetDisplayColor(i, OFF);
             }
            }
        }
      }
      if(TimesPlayedTotal >= 6)
      {
        for (i = 0; i <= 24; i++)
        {
           SetDisplayColor(i, OFF);
          delay  (100);
        }
        for (i = 24; i >= 0; i--)
        {
           SetDisplayColor(i, GREEN);
          delay(100);
                }
         SwitchGame(0);
      }
  }
}

int  RandomColumn1And2()  {
  int  led  =random(1, 9);  // Random no. 1-8
  switch(led) {
    case 1:
      led = 1;
      break;
    case 2:
      led = 2;
      break;
    case 3:
      led = 7;
      break;
    case 4:
      led = 8;
      break;
    case 5:
      led = 13;
      break;
    case 6:
      led = 14;
      break;
    case 7:
      led = 19;
      break;
    case 8:
      led = 20;
```

```
      break;
    }
  return  led;
}

int  RandomColumn3And4()  {
  int  led  =random(1, 9);  // Random  no.  1-8
  switch(led) {
    case  1:
      led = 3;
      break;
    case  2:
      led = 4;
      break;
    case  3:
      led = 9;
      break;
    case  4:
      led = 10;
      break;
    case  5:
      led = 15;
      break;
    case  6:
      led = 16;
      break;
    case  7:
      led = 21;
      break;
    case  8:
      led = 22;
      break;
    }
  return  led;
}

int  RandomColumn5And6()  {
  int  led  =random(1, 9);  // Random  no.  1-9
  switch(led) {
    case  1:
      led = 5;
      break;
    case  2:
      led = 6;
      break;
    case  3:
      led = 11;
      break;
    case  4:
      led = 12;
      break;
    case  5:
```

```
        led = 17;
        break;
      case 6:
        led = 18;
        break;
      case 7:
        led = 23;
        break;
      case 8:
        led = 24;
        break;
    }
  return led;
}
```

```
/*              Appendiks C-IV: Arduino-softwarens kildekode              */
/*                                  SLAVE                                 */
#define       BAUD       115200// Baud rate for all RS232-communication
#define        MAXINP         30// Max chars in input from Master

int      I         =         0;// Used for reading serial input from the Master
char     InpFromMaster[MAXINP];// Used for reading input from the Master
void setup() {
  Serial1.begin(BAUD);

  pinMode(25, OUTPUT);
  digitalWrite(25, HIGH);

  // PWM outputs
  int p;
  for (p = 2; p <= 13; p++) {
    //pinMode(p, OUTPUT);        // Init the port as PWM output <- IKKE NØDVEND
    analogWrite(p,         0);// Turn off the display
  }
}

// Sets the color of a display
void  SetDisplayColor(int  dispNo,int color) {
  analogWrite(dispNo+1, color);  // Add one, as the pin numbers are 2-13
}

// Collects, assembles and processes any serial input from the Master
void  CollectSerialInput()  {
  // If there's something to read from the Master
  while (Serial1.available()) {
      InpFromMaster[I] =Serial1.read();
    // If there's a line end signal, process the input
    if  (InpFromMaster[I]  =='\n') {

      // Read the input
      // If its a set display-color command (D01,255 for display 01 color 255
      if  (InpFromMaster[0]  =='D') {
        // Fetch the display no
        int disp = 10*(int(InpFromMaster[1])-48);
         disp +=int(InpFromMaster[2])-48;

        // Get the color no
        // color should be changed to a byte
        int color = 100*(int(InpFromMaster[4])-48);
        color += 10*(int(InpFromMaster[5])-48);
         color +=int(InpFromMaster[6])-48;

         SetDisplayColor(disp, color);
      }

      // Flush the string array
       InpFromMaster[0] ='0'; // Numeric zero, not the letter
```

```
        I = -1;
      }
      I++;
    }
}

void loop() {
    CollectSerialInput();
}
```