# Gesamt-Software

## Anpassungen

Bei der Zusammenführung der Arbeitspakete AP3. bis AP8. wurden Änderungen vorgenommen.   
Das Programm besitzt nun drei Unterprogramme («Start\_Taster()»,«Blinken()»,«Zuruecksetzung()»). Diese verkürzen den Programm-Code.

|  |  |
| --- | --- |
| /\* Projektnamen: Reaktionsspiel  \* Dateinamen: Gesamtprogramm\_Reaktionsspiel.ino  \* Entwickler: T. Ammann, I. Christodulakis  \* Datum: 29.05.2017  \* Ort: 6301 Zug, CH  \* Entwicklungsschritt: Zusammenführung, 29.05.2017  \* Formatierung, 29.05.2017  \* Erweiterung um Unterprogramm  \* "Start\_Taster", "Blinken" &  \* "Zuruecksetzung"  \* Letztes Aenderungsdatum: 29.05.2017  \*/  /\* Variablen ------------------------------------------------------------------- \*/  **bool** St\_Si; // Start-Signal  **char** EingangSpieler[2],  Spieler[2],  Zms[2],  Spielersignal;  **char**  Pu\_Za1 = 0, // Punktzahl1  Pu\_Za2 = 0; // Punktzahl2  **char** a, // fuer for-Schleifen {  b,  c,  x,  y; // }  **unsigned long** Anfangszeit, // Anfangszeit der Zeitmessung  Endzeit, // Endzeit der Zeitmessung  Zeit,  Zufallszahl, // zufaellig generierte Zahl  Re\_Ze1, // Reaktionszeit1  Re\_Ze2, // Reaktionszeit2  Zufallszeit; // zufaellig generierte Zeit (0 <= Wert < 4001)  /\* -------------------------------------------------------------------------------- \*/ | Die Variable «St\_Si» ist vom Datentyp «bool», da sie nur die Werte 0 & 1 besitzen kann.  Die Variablen vom Daten-Typ «char», da sie nur einstellige Dezimal-Werte annehmen können.  Die Variablen vom Datentyp «unsigned long» speichern Zeiten, welche von der Funktion «millis()» bestimmt werden. «millis()» gibt den Wert in «unsigned long» aus. |

|  |  |
| --- | --- |
| /\* Konstanten ---------------------------------------------------------------- \*/  **const char** S\_S = 3, // Eingaenge {  Q\_S1 = 4,  Q\_S2 = 5, // }  P\_S1 = 6, // Ausgaenge {  P\_S2 = 7,  O\_M1 = 8,  O\_M2 = 9,  R\_S = 10; // }  /\* -------------------------------------------------------------------------------- \*/  /\* Setup ----------------------------------------------------------------------- \*/  **void setup()**{  /\* Eingaenge ----------------------------------------------------------------- \*/  **pinMode**(S\_S, INPUT);  **pinMode**(Q\_S1, INPUT);  **pinMode**(Q\_S2, INPUT);  /\* Ausgaenge ----------------------------------------------------------------- \*/  **pinMode**(P\_S1, OUTPUT);  **pinMode**(P\_S2, OUTPUT);  **pinMode**(O\_M1, OUTPUT);  **pinMode**(O\_M2, OUTPUT);  **pinMode**(R\_S, OUTPUT);  /\* Pin fuer Zufallszahl ------------------------------------------------------ \*/  **randomSeed**(**analogRead**(0)); // analogPin0 nicht anschliessen!  }  /\* -------------------------------------------------------------------------------- \*/ | Die Konstanten sind im Datentyp «char», da sie einfache Werte (Charakter) speichern.  Im Setup werden die Ein- und die Ausgänge definiert.  Mit der Funktion «randomSeed()» wird die Referenz für die Zufallszahlgenerierung definiert. |

|  |  |
| --- | --- |
| /\* Unterprogrammdeklaration ------------------------------------------ \*/  // Start-Signal einlesen  **void** Start\_Taster(**char** Start\_Pin);  // Optische Meldung ein- & ausschalten  **void** Blinken(**void**);  //Anzeigen zuruecksetzten  **void** Zuruecksetzung(**char** Punktzahl\_Pin1, **char** Punktzahl\_Pin2, **char** Reset\_Pin);  /\* -------------------------------------------------------------------------------- \*/  /\* Hauptprogramm =================================== \*/  **void loop()**{  Zuruecksetzung(P\_S1, P\_S2, R\_S);  /\* AP3. ------------------------------------------------------------------------- \*/  Start\_Taster(S\_S);  **for**(a = 0; a < 3; a++){ // Optische Meldung ausgeben  Blinken();  }  /\* -------------------------------------------------------------------------------- \*/  /\* AP7. ------------------------------------------------------------------------- \*/  **do**{  /\* AP4. ------------------------------------------------------------------------- \*/  Zufallszahl = **random**(100);  **if**(Zufallszahl %2 == 0){  EingangSpieler[0] = Q\_S1;  EingangSpieler[1] = Q\_S2;  Zms[0] = O\_M1;  Zms[1] = O\_M2;  Spieler[0] = 1;  Spieler[1] = 2;  }  **else**{  EingangSpieler[0] = Q\_S2;  EingangSpieler[1] = Q\_S1;  Zms[0] = O\_M2;  Zms[1] = O\_M1;  Spieler[0] = 2;  Spieler[1] = 1;  }  /\* -------------------------------------------------------------------------------- \*/ | Die Unterprogramme «Start\_Taster», «Blinken» & «Zuruecksetzung» werden deklariert.  Diese Funktionen werden im Programm mehrmals verwendet. Um den benötigten Speicher zu verkleinern werden Unterprogramme erstellt.  Das Hauptprogramm befindet sich in der Funktion «loop()». Diese wird ständig wiederholt.  AP3. führt «Start\_Taster» aus. Danach wird «Blinken» drei mal ausgeführt.  AP4. generiert zunächst eine Zufallszahl zwischen 0 und 99.  Abhängig davon, ob die Zahl gerade oder ungerade ist werden die Arrays «EingangSpieler», «Zms» & «Spieler» einem Wert zugeordnet. |

|  |  |
| --- | --- |
| /\* AP5. ------------------------------------------------------------------------- \*/  **for**(x = 0; x < 2; x++){  Zufallszeit = **random**(4001);  Anfangszeit = **millis()**;  **do**{  Spielersignal = **digitalRead**(EingangSpieler[x]);  Zeit = **millis()**;  **if**(Zeit >= Anfangszeit + Zufallszeit+ 1000){  **digitalWrite**(Zms[x], HIGH);  }  }**while**(Spielersignal == 0);  Endzeit = **millis()**;  **if**(Endzeit >= Anfangszeit+ Zufallszeit + 1000){  Reaktionszeit = Endzeit – Anfangszeit – Zufallszeit – 1000;  }  **else**{  Reaktionszeit = 10000;  **for**(c = 0; c <2; c++){  Blinken();  }  }  **if**(Spieler[x] == 1){  Re\_Ze1 = Reaktionszeit;  }  **else**{  Re\_Ze2 = Reaktionszeit;  }  **digitalWrite**(Zms[x], LOW);  }  /\* -------------------------------------------------------------------------------- \*/ | AP5. führt die Zeitmessung durch. Diese wird zwei Mal wiederholt.  Zunächst wird eine Zufällige Zahl zwischen 0 und 4000 generiert.  Danach wird der Messvorgang durchgeführt..  Zuletzt wird die Reaktionszeit dem entsprechenden Spieler zugeordnet. |

|  |  |
| --- | --- |
| /\* AP6. ------------------------------------------------------------------------- \*/  **if**(Re\_Ze1 > Re\_Ze2){  **digitalWrite**(P\_S2, HIGH);  **delay**(100);  **digitalWrite**(P\_S2, LOW);  Pu\_Za2++;  }  **else** **if**(Re\_Ze2 > Re\_Ze1){  **digitalWrite**(P\_S1, HIGH);  **delay**(100);  **digitalWrite**(P\_S1, LOW);  Pu\_Za1++;  }  /\* -------------------------------------------------------------------------------- \*/  /\* AP7. ------------------------------------------------------------------------- \*/  }**while**(Pu\_Za1 < 9 && Pu\_Za2 < 9);  **if**(Pu\_Za1 == 9){  **digitalWrite**(O\_M1, HIGH);  }  **else**{  **digitalWrite**(O\_M2, HIGH);  }  /\* -------------------------------------------------------------------------------- \*/  /\* AP8. ------------------------------------------------------------------------- \*/  Start\_Taster(S\_S);  **digitalWrite**(O\_M1, LOW);  **digitalWrite**(O\_M2, LOW);  Zuruecksetzung(P\_S1, P\_S2, R\_S);  Pu\_Za1 = 0; // Punktzahl1 zuruecksetzen  Pu\_Za2 = 0; // Punktzahl2 zuruecksetzen  /\* -------------------------------------------------------------------------------- \*/  }  /\* ================================================= \*/ | AP6. wertet die Reaktionszeiten aus und vergibt dem Spieler mit der kleineren Reaktionszeit einen Punkt.  AP7. führt den gesamten Ablauf durch, bis einer der Spieler neun Punkte besitzt.  AP8. führt zunächst «Start\_Taster» aus. Danach werden die Ausgänge «O\_M1» & «O\_M2» auf logisch LOW gesetzt. Anschliessend wird «Zuruecksetzung» ausgeführt und die Punkte der Spieler werden genullt. |

|  |  |
| --- | --- |
| /\* Unterprogrammdefinition --------------------------------------------- \*/  // Start-Signal einlesen  **void** Start\_Taster(**char** Start\_Pin){  **do**{  St\_Si = **digitalRead**(Start\_Pin);  **delay**(50);  }**while**(St\_Si == 0);  }  // Optische Meldung ein- & ausschalten  **void** Blinke(**void**){  **for**(b = 1; b > -1; b--){  **delay**(500);  **digitalWrite**(O\_M1, b);  **digitalWrite**(O\_M2, b);  }  }  // Anzeigen zuruecksetzen  **void** Zuruecksetztung(**char** Punktzahl\_Pin1, **char** Punktzahl\_Pin2, **cha**r Reset\_Pin){  **digitalWrite**(Reset\_Pin, LOW);  **for**(y = 1; y > -1; y--){  **delay**(100);  **digitalWrite**(Punktzahl\_Pin1, y);  **digitalWrite**(Punktzahl\_Pin2, y);  }  **digitalWrite**(Reset\_Pin, HIGH);  }  /\* -------------------------------------------------------------------------------- \*/ | «Start\_Taster» liest den Wert an «Start\_Pin» solange ein, solange der Wert null entspricht. Danach wird das Programm verlassen.  «Blinken» schaltet die Ausgänge «O\_M1» und «O\_M2» zunächst ein, wartet 500 Milli-Sekunden und schaltet diese wieder aus.  «Zuruecksetzung» setzt den Ausgang «Reset\_Pin» auf logisch LOW. Danach wird aus den Ausgängen «Punktzahl\_Pin1» & «Punktzahl\_Pin2» ein HIGH-Impuls ausgegeben. Zuletzt wird der «Reset\_Pin» auf logisch HIGH gesetzt. |