Praktikum 5./12./18./19. Januar 2012

Jörn Loviscach

Versionsstand: 3. Januar 2012, 19:20

In diesem Praktikum soll ein Geschicklichkeitsspiel programmiert werden.

Das C-Projekt basiert auf der kleinen Funktionsbibliothek, die aus `display_joystick.c` und `display_joystick.h` besteht. Von besonderem Interesse sind diese Funktionen:

```
void initialize(void);
unsigned int readAnalog(int channel);
void writeString(char s[]);
void waitMilliseconds(unsigned int duration);
```

Die Aufrufe `readAnalog(0)` und `readAnalog(1)` geben dabei die x- und y-Koordinaten des Joysticks als Zahlen von 0 bis 1023 zurück.

Schließen Sie das Display (Link zur Anschlussbelegung) so an das LaunchPad an: jeweils Versorgungsspannung und Masse verbinden, Pin 1.0 an RS, Pin 1.1 an E, Pins 1.4 bis 1.7 an die Datenleitungen D4 bis D7. Verbinden Sie die Schleifer des Joysticks mit Pin 1.2 und Pin 1.3.

Lassen Sie den Aufbau sicherheitshalber kontrollieren, bevor Sie ihn mit dem PC verbinden. Schreiben Sie ein kleines Programm, um sich von der Funktionsfähigkeit zu überzeugen. Dieses Programm soll in einer Endlosschleife die Koordinaten einlesen und anzeigen.

Das Geschicklichkeitspiel soll dann so funktionieren: Der Joystick soll quasi als Lenkrad (x-Achse) und als Gaspedal (y-Achse) dienen. Das Ziel ist, einen Parcours möglichst schnell ohne Fehler abzufahren. Auf dem Display wandert eine Zielposition hin und her, markiert durch den Buchstaben `T`. Die aktuelle x-Koordinate des Joysticks wird durch `> o <` angezeigt. Dieses Buchstabenmuster wird an den Rändern des Displays abgeschnitten; zum Beispiel ganz links erscheint nur `o <`. Im optimalen Fall steht der Joystick perfekt auf der Zielposition. Dann erscheint `> T <` im Display.

Eine Datensammlung wie die folgende gibt an, wie die Zielposition wandern soll. Dabei gibt die erste Zahl die Zeit (beliebige Einheiten) an, nach der die Zielposition auf den als zweites angegebenen Wert springen soll (0 bis 7 für die acht auf dem

Display möglichen Positionen). Ein Wert von −1 bei der Position kennzeichnet das Ende; dort soll zum Anfang gesprungen werden.

```
static const Waypoint tour[] =
{
   {   0, 4},
   { 100, 5},
   { 300, 6},
   { 400, 7},
   { 500, 6},
   { 550, 5},
   { 600, 4},
   { 650, 3},
   { 700, 2},
   { 750, 1},
   { 800, 0},
   { 900, 1},
   {1000, 2},
   {1100, 3},
   {1200, 4},
   {1300,-1}
};
```

Je nach verbleibender Zeit erweitern Sie das Programm zum Beispiel so:

- Nach einer festen Zahl von Runden werden die Gesamtdauer und die Zahl an Fehlern (starke Abweichungen von Zielposition und tatsächlicher Position) angezeigt.
- In der Liste mit den vorgegebenen Punkten sind nicht alle Zwischenschritte anzugeben; vielmehr wird linear interpoliert.