Universität Siegen

Embedded Control Laboratory

Ball on inclined plane

Report of lab excercise 3

WS 2016/17

Group 11

901510 David Schlosser
918781 Stefan Schmidt

Inhaltsverzeichnis

1 Bericht

1.1 Einleitung

In diesem Versuch haben wir vor, die in Lab 2 errechneten Werte für den PID-Regler in den vorhandenen Code einzufügen und so ein funktionierendes Experiment zu haben. Zusätzlich mussten wir die Werte für $MIDPOS$ und $DELTA$ berechnen und den Willkommenstext auf der LCD-Anzeige ändern.

1.1.1 Verwendung von OCR3A

OCR3A ist das Timer Output Compare Register, ...
TODO

1.1.2 Berechnung von MIDPOS

Der richtige MIDPOS Wert gibt an, wann die Fläche vollkommen horizontal steht. Zur Berechnung von MIDPOS benutzten wir folgende Formel:

$$MIDPOS = Frequenz/Prescaler/1000 * Pulslaenge \quad (1)$$

Die Frequenz war vorgegeben mit $16Mhz$. Der Prescaler-Wert wurde am Port TCCR3B mit 64 festgelegt und konnte mit Hilfe des Datenblattes aus dem Code abgelesen werden. Die Pulslänge bei horizontaler Lage wurde in der Aufgabenstellung erwähnt und lag bei $1,5ms$.
Das Ergebnis war $MIDPOS = 375$, allerdings war die Fläche nicht vollständig horizontal, nach kurzem Anpassen kamen wir auf den Wert $MIDPOS = 355$.

1.1.3 Berechnung von DELTA

DELTA ist die Differenz zwischen der maximalen linken oder rechten Position der Fläche und der mittleren Position (MIDPOS). Für die Berechnung des Wertes von DELTA verwendeten wir die folgende Formel:

$$DELTA = (Frequenz/Prescaler/1000 * Pulslaenge) - MIDPOS \quad (2)$$

Für den Wert der Pulslänge mussten wir diesmal allerdings $2ms$ verwenden da wir den Wert für die maximale rechte Position der Fläche benötigten. Von diesem Wert zogen wir $MIDPOS$ ab um die Differenz zu erhalten. Nun mussten wir wieder den Wert leicht anpassen und kamen auf das Ergebnis $Delta = 480$.

1.1.4 Willkommens-Anzeige

Die Willkommensnachricht auf der LCD Anzeige konnten wir in der Datei $HMI.c$ in Zeile 260 einfach ändern.

1.1.5 PID Implementation

Die Implementierung des PID-Reglers erfolgte komplett in der Datei $PID.c$. Die Formel die wir verwendeten für den Wert des PID-Reglers lautete:

$$y = (proportional * error) + (integral * errorSum * sampleTime) + (derivative * (error * errorLast)/sampleTime) \quad (3)$$

Die Werte für $proportional$, $integral$ und $derivative$ konnten wegen Abweichung im realen Versuch leider nicht übernommen werden. Nach längerem kalibrieren kamen wir auf die Werte $proportional = 0,19$, $integral = 0,018$ und $derivative = 0,51$.
Als $sampleTime$ verwendeten wir die in der Aufgabenstellung erwähnte Signalperiode von $18ms$ $(0,018s)$. $error$ berechneten wir als Differenz zwischen der Zielposition des Balls und der aktuellen Position.

1.1.6 Vergleich der Ergebnisse mit Versuch 2

Im Vergleich zu Versuch 2 benötigten wir beim eigentlichen Modell vollkommen andere PID-Werte. Ein möglicher Grund dafür ist, dass wir in Versuch 2 ein kontinuierliches System simuliert haben, in Versuch 3 aber mit einem diskreten System gearbeitet haben. TODO

1.1.7 Queues und Tasks

Es gibt die drei Queues QueueTaster, QueueSensor und QueueServo sowie die sechs Tasks vSensor, vIO_SRAM_to_LCD, vTaster, vHMI, vServo und vPID.

QueueTaster
Diese Queue erhält Input-Daten von *vTaster*, diese werden dann weiter gegeben an *vHMI* und an *change_para*.
QueueSensor
QueueSensor empfängt die Positionsdaten des Balls von *vSensor* und sendet diese an *vPID*.
QueueServo
QueueServo verwaltet den PID-Wert der in *vPID* berechnet wurde und gibt sie an *vServo* weiter.
vSensor
Dieser Task berechnet die aktuelle Ballposition mit Hilfe der Sensoren und sendet die Position an *QueueSensor*.
vIO_SRAM_to_LCD
Dieser Task kopiert Daten vom RAM zum LCD und positioniert den Cursor im Menü.
vTaster
vTaster liest Input Daten an den Pins und sendet diese Parameter an *QueueTaster*.
vHMI
Dieser Task steuert/verwaltet die Menüs und Funktionen, welche auf dem LCD dargestellt werden, überprüft welcher Parameter zur Zeit in *Queuetaster* ist und verwendet diesen zur Menüsteuerung.
vServo
Dieser Task berechnet den Vergleichswert des Servosignals und setzt diesen Wert auf den Port *OCR3A*. *vServo* erhält den Wert *Beta* der für die Berechnung benötigt wird aus *QueueServo*.
vPID
Dieser Task holt sich die aktuelle Ballposition aus *QueueSensor* und berech-

net damit den aktuellen PID-Wert.

1.1.8 Inter Task Kommunikation

TODO

1.2 Ergebnis

Mit unseren aktualisierten PID-Werten ist es uns gelungen den Ball in weniger als zehn Sekunden an der gewünschten Stelle stehen bleiben zu lassen.

1.3 Fazit

TODO