Universität zu Lübeck

Institut für Technische Informatik

Direktor: Prof. Dr.-Ing. Mladen Berekovic

Studienbegleitende Fachprüfung im Rahmen der
Bachelorprüfung im Studiengang
Robotik und Autonome Systeme / Informatik /
Medieninformatik / Biophysik
Leistungszertifikat Typ A/B/A/A
WS 2021/2022

Lehrmodul: Einführung in die Robotik und Automation

Prüfer: Prof. Dr.-Ing. Heiko Hamann

30.03.2022

Aufgabe	**1**	**2**	**3**	Σ
Punkte	30	20	30	80

1 SPS: Speicherprogrammierbare Steuerungen (30 Punkte)

In den folgenden Aufgaben implementieren Sie die Steuerung eines intelligenten Mikrowellenofens schrittweise unter Verwendung einer Speicherprogrammierbaren Steuerung. Die Teilaufgaben ergeben zusammen das Gesamtverhalten der Maschine, sind jedoch unabhängig voneinander bearbeitbar.

Variable	Typ	Funktion
tStart	Eingang Taster-Schließer	Start des Zubereitungsvorgangs
tStop	Eingang Taster-Schließer	Abbruch des Zubereitungsvorgangs
tHoch	Eingang Taster-Schließer	Auswahl Inkrementieren
tRunter	Eingang Taster-Schließer	Auswahl Dekrementieren
sTür	Eingang Schalter-Öffner	Türschalter erkennt offene Tür
iTSens	Eingang DInt	Aktuelle Temperatur in °C
iScanner	Eingang DInt	Barcode des gescannten Produkts
aHeizen	Ausgang	Steuert das Heizelement
aLüfter	Ausgang	Steuert den Lüfter
aMikro	Ausgang	Steuert die Mikrowellenröhre
vTDisp	Variable - DInt	Eingestellte Temperatur in °C
vDDisp	Variable - Time	Eingestellte Dauer in Sekunden
vBereit	Variable - Bool	Start des Spülvorgangs
vEnable	Variable - Bool	Gibt manuelle Eingabe frei
vErkannt	Variable - Bool	Gibt an, ob ein Produkt erkannt wurde
vTMan	Variable - DInt	Manuelle Temperatur in °C
vTErk	Variable - DInt	Erkannte Temperatur in °C
vDMan	Variable - Time	Manuelle Dauer in Sekunden
vDErk	Variable - Time	Erkannte Dauer in Sekunden

Tabelle 1: Global verfügbare Variablen

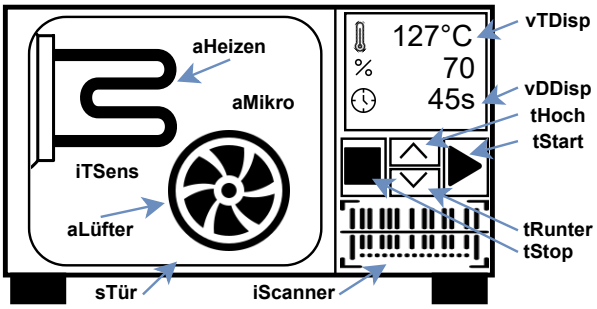

Abbildung 1: Schematischer Aufbau des intelligenten Mikrowellenofens

Der intelligente Mikrowellenofen besteht aus einem Heizelement **aHeizen**, einer Mikrowellenröhre **aMikro**, einem Lüfter **aLüfter**, sowie einem Barcodescanner **iScanner**, einem Bedienfeld und einem Display. Das Display zeigt die Temperatur, die Leistung der Mikrowellenröhre und die Dauer des Zubereitungsvorgangs an. Der Mikrowellenofen kann mittels des Scanners **iScanner** den Barcode eines Produktes lesen und anhand einer Datenbank die Einstellungen für den Zubereitungsvorgang automatisch vornehmen. Da nicht alle Produkte einen Barcode haben, hat der Mikrowellenofen ein Bedienfeld, über das sich manuell die Zeit, Temperatur und Leistung des Zubereitungsvorgangs einstellen lassen. Zur Sicherheit verfügt der Mikrowellenofen über einen Türschalter **sTür**, der das Öffnen während des Betriebs erkennt. Die Benutzung des Mikrowellenofens erfolgt wie folgt: Es wird ein Produkt vor den Scanner gehalten, die Temperatur, Leistung und Dauer werden automatisch gelesen. Alternativ wird über das Drücken der Tasten **tHoch** und **tRunter** zunächst die Temperatur eingestellt. Durch Drücken der Start-Taste **tStart**, lässt sich danach die Leistung der Mikrowellenröhre über die Tasten **tHoch** und **tRunter** einstellen und nach einem weiteren Drücken auf **tStart** die Zubereitungsdauer. Der Zubereitungsvorgang wird über den Knopf **tStart** gestartet, nachdem alle Einstellungen gemacht wurden. Der Einstellvorgang, sowie die laufende Zubereitung, lassen sich über den Knopf **tStop** abbrechen. Ein Öffnen der Tür (**sTür**) bricht den Zubereitungsvorgang ab. In den folgenden Teilaufgaben wird das jeweils zu implementierende Verhalten genauer erläutert.

Hinweis: Verwenden Sie für KOP nur einfache Kontakte und Spulen (kein SET/RESET). Sollten Sie in FUP mit Flipflops arbeiten, **geben Sie explizit den Typ an**, indem Sie den dominanten Eingang mit einer 1 markieren (z.B. R1 für Rücksetzdominant). Selbiges gilt für Timerbausteine.

Variablen: Die Ihnen zur Verfügung stehenden globalen Variablen sind in Tabelle 1 definiert. Sie können in den einzelnen Teilaufgaben ggf. weitere lokale Variablen verwenden ohne diese extra zu deklarieren.

(a) Entwerfen Sie eine Schaltung, die erkennt, ob der Barcode eines Produkts in der Produkt-Datenbank hinterlegt und das Produkt für den Mikrowellenofen geeignet ist.
Stellen Sie dafür zunächst eine Wahrheitstabelle auf, in der Variable **vErkannt** ein „True“ zugewiesen wird, wenn der der Barcode in der Datenbank und gültig ist. Ein Barcode erfüllt die Kriterien, wenn in der **Binärdarstellung des Barcodes die Quersumme der unteren vier Bits von iScanner** $\geq$ **2** ist.
Auf die einzelnen Bits von **iScanner** kann über die Variablen **S3 bis S0** zugegriffen werden, wobei **S3** dem höchsten (links) und **S0** dem niedrigsten Bit (rechts) entspricht.
Beispiel: Quersumme(„1101“) = 3 → **vErkannt** = „True“
Minimieren Sie im Anschluss Ihre Wahrheitstabelle mittels eines KV-Diagramms und geben die resultierende DNF an.
Erstellen Sie zuletzt eine Schaltung in **KOP**, welche der Variable **vErkannt** ein „True“ zuweist, wenn der Barcode des Produkts in der Datenbank und gültig ist.

(b) Entwerfen Sie eine Schaltung in **FUP**, welche die manuelle Einstellung des Zubereitungsvorgangs umsetzt. Wenn das Enable-Signal **vEnable** gesetzt ist, sollen die Variablen **vDMan** und **vTMan** über die Taster **tHoch** und **tRunter** eingestellt werden, wobei zuerst die Temperatur **vTMan** eingestellt werden soll. Nach einem Druck auf den Taster **tStart** sollen die Taster die **vDMan** einstellen. Mit einem Druck auf **tHoch** wird der einzustellende Wert um Eins inkrementiert und mit einem Druck auf **tRunter** um Eins dekrementiert. Wenn der Stop Taster **tStop** gedrückt wird, soll der Einstellvorgang zurückgesetzt und die Variablen **vTMan** und **vDMan** zurück auf Null gesetzt werden. Ist das Signal **vEnable** nicht gesetzt, soll die Schaltung alle Eingaben ignorieren.
Es genügt, einen der beiden Zähler zu implementieren und die Verwendung des zweiten zu beschreiben. Heben Sie dabei die Unterschiede hervor.

(c) Implementieren Sie eine drahtbruchsichere Selbsthalteschaltung in **FUP oder KOP**, welche die Variable **vBereit** steuert. Der Taster **tStart** setzt die Variable **vBereit**. Sobald die Tür (**sTür**) geöffnet oder **tStop** gedrückt wird, wird **vBereit** zurückgesetzt. Ein Setzen von **vBereit** bei offener Tür oder gedrückter **tStop** Taste muss verhindert werden.

Wenn Sie FUP als Sprache verwenden, dürfen in dieser Teilaufgabe nur UND- und ODER-Gatter, sowie Negationen verwendet werden.

(d) Implementieren Sie in **FUP** eine einfache Lüftersteuerung. Wenn die Temperatur **iTSens** im Mikrowellenofen über der eingestellten Temperatur **vTDisp** liegt, soll der Lüfter **aLüfter** eingeschaltet werden, bis die Temperatur im Mikrowellenofen 5 °C unter der eingestellten Temperatur liegt.

(e) In Form einer Ablaufsteuerung (**AS/Graph**) soll das Verhalten des Mikrowellenofens realisiert werden: Der Mikrowellenofen befindet sich im Initialzustand, in dem alle Aktoren ausgeschaltet werden und die Eingabe **vEnable** deaktiviert ist. Im Initialzustand wird gewartet, bis entweder ein Produkt vom Scanner erkannt wird (**vErkannt**) oder manuell ein Zubereitungsvorgang durch Drücken von **tStart** gestartet wird. Während des manuellen Einstellens (**vEnable**=„True“) oder nach dem Erkennen eines Produktes sollen Temperatur (**vTErk**, **vTMan**) und Dauer (**vDErk**, **vDMan**) auf dem Display (**vTDisp**, **vDDisp**) angezeigt werden. Nachdem der Zubereitungsvorgang mit einem Druck auf **tStart** gestartet wird, soll das Heizelement **aHeizen** für die Dauer **vDDisp** eingeschaltet werden, während die Mikrowellenröhre **aMikro** eingeschaltet ist, bis die eingestellte Temperatur **vTDisp** erreicht ist.
Wird während der Zubereitung die Stop-Taste **tStop** gedrückt oder die Tür (**sTür**) geöffnet, wird der Zubereitungsvorgang abgebrochen und zurück in den Initialzustand gesprungen.

Hinweise: Sie dürfen zur besseren Lesbarkeit Transitionsbedingungen anstelle von FUP und KOP auch als logischen Ausdruck angeben.
In Zuständen können Variablenzuweisungen durchgeführt werden.
N | A:=10 weist beispielsweise der Variable A den Wert 10 zu.

2 Regelungstechnik (20 Punkte)

(a) Ihre Aufgabe ist es für ein UAV (Unmanned Aerial Vehicle) mit vier Rotoren, welche alle eine vertikale Schubrichtung haben, eine Regelung für die Flughöhe zu entwerfen. Zeichnen Sie in einem ersten Schritt ein Blockschaltdiagramm für alle wichtigen Komponenten dieses Regelproblems. Überlegen Sie sich für jedes Teilsignal geeignete Einheiten und zeichnen Sie diese in das Blockschaltdiagramm.

(b) Nennen Sei drei Vorteile für die Verwendung eines PID-Reglers für dieses Projekt.

(c) Erläutern Sie für das Szenario mit der Höhenregelung eines UAVs welche Problematik auftritt, wenn lediglich ein Proportional-Regler verwendet wird.

(d) Die meisten Aktoren in realen Systemen erreichen ab einem gewissen Stellwert einen saturierten Zustand. Die Rotoren eines UAVs haben zum Beispiel eine maximale Drehzahl. Auch bei einem höheren Stellwert, werden sich die Rotoren also nicht schneller als ihre maximale Drehzahl bewegen können. Beschreiben Sie, welche Probleme beim Einsatz eines Integral-Reglers für ein System mit Aktoren, welche Saturation erreichen können, auftreten können. Wie könnte dieses Problem gelöst werden?

(e) Sie bekommen die Aufgabe, Parameter für einen PID-Regler zu finden. Bei der Regelstrecke handelt es sich um eine PT1-Strecke mit Totzeit, welche zu Schwingungen neigt. Welche strukturierte Herangehensweise zur Einstellung der Regelparameter kennen Sie für diese Regelstrecke? Erklären Sie das Verfahren und welche Eigenschaften das Regelverhalten mit den so gefundenen Regelparametern aufweist.

(f) In Abbildung 2 ist das Eingangssignal x für einen PID-Regler mit folgenden Parametern gegeben: $K_P = 1{,}5$, $K_I = 1$ und $K_D = 2$.
Zeichnen Sie jeweils die Ausgangsfunktionen für den P-, I- sowie für den D-Anteil.

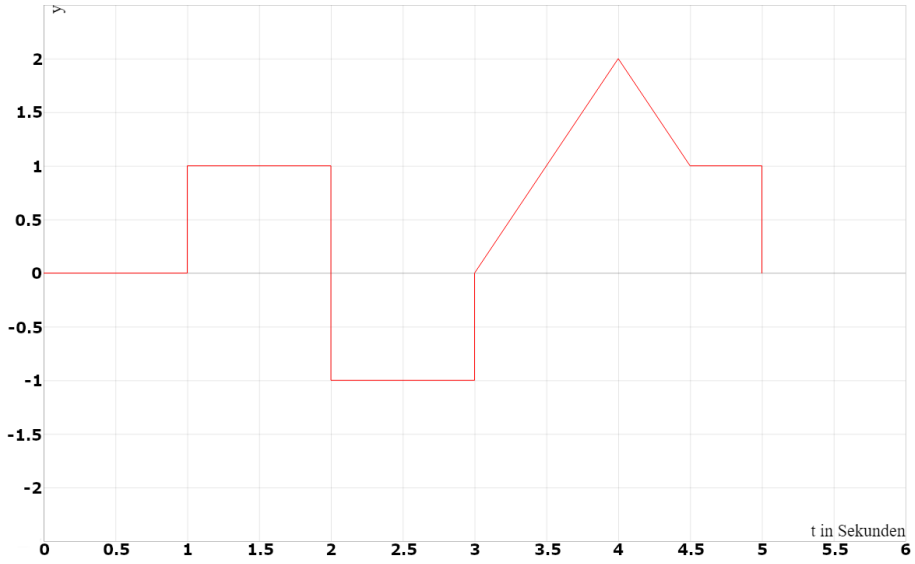

Abbildung 2: Eingangssignal für einen PID-Regler

3 Mobile Robotik (30 Punkte)

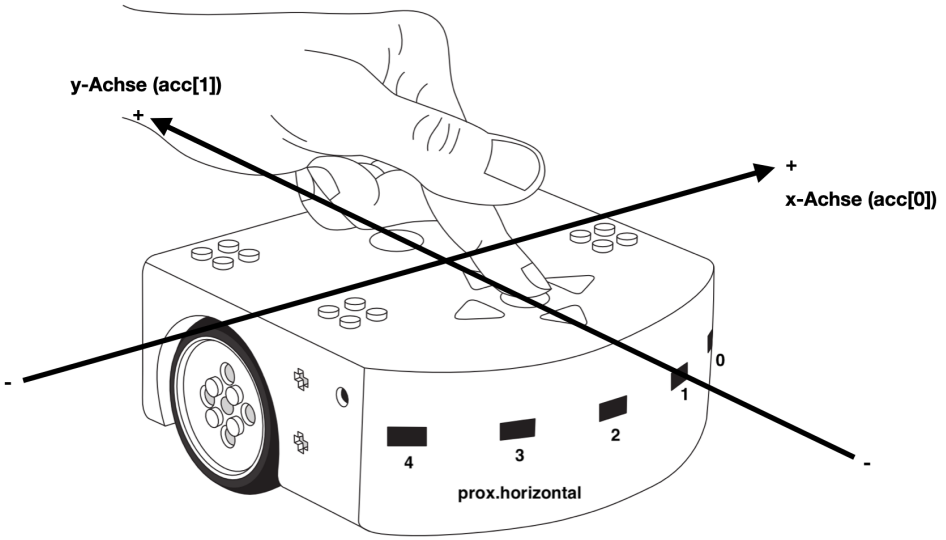

Abbildung 3: Thymio II mit den vorderen horizontalen IR-Sensoren und dem Beschleunigungssensor.

In dieser Aufgabe müssen Sie ein Programm für den Thymio II mit der Programmiersprache Aseba implementieren.

Der Roboter befindet sich auf einer Schräge, soll bergauf fahren und dabei möglichen Hinternissen ausweichen. Um herauszufinden in welche Richtung es bergauf geht, sollen Sie den eingebauten Beschleunigungssensor (Accelerometer) verwenden. Die Messachsen des Sensors können Sie zur Veranschaulichung in Abbildung 3 sehen. Genauere Informationen zur Verwendung erhalten Sie in der dazugehörigen Teilaufgabe.

Die Basissteuerung besteht aus den folgenden Status:

- `STOP`: Der Roboter steht still und bewegt sich nicht. (Anfangszustand)
- `REORIENT`: Der Roboter dreht sich in einer von Ihnen gewählten Richtung auf der Stelle bis er bergauf ausgerichtet ist.
- `GO`: Der Roboter fährt bergauf. Hierbei soll die Richtung anhand der Werte des Beschleunigungssensors angepasst werden.

Zusätzlich sollen Sie ein simples Ausweichverhalten implementieren:

- `BACK`: Der Roboter hat ein Hindernis entdeckt und fährt für 0,5 Sekunden rückwärts.
- `TURN`: Nachdem der Roboter rückwärts gefahren ist, dreht er sich für 1,2 Sekunden in einer von Ihnen gewählten Richtung auf der Stelle.
- `AVOID`: Nachdem der Roboter sich gedreht hat, fährt er für 2 Sekunden geradeaus.
- `TURNBACK`: Final dreht der Roboter sich ebenfalls für 1,2 Sekunden zurück, bevor er wieder in den Status `REORIENT` wechselt.

Weitere Details zur Implementierung finden Sie in den entsprechenden Teilaufgaben.

Wichtige Hinweise zur Programmierung

- Programmieren Sie den Thymio mit der Programmiersprache Aseba (Text-Programmierung).
- Halten Sie sich an die im Text vorgegebenen Spezifikationen des Thymios.
- Verwenden Sie vorgegebene Konstanten und Variablen in Ihrer Implementierung.
- Verwenden Sie pro Teilaufgabe maximal ein Ereignis (Event).
- Wählen Sie für die Teilaufgabe passende Ereignisse, wobei Ereignisse nicht mehrfach verwendet werden dürfen.
- Wählen Sie für nicht vorgegebene Werte selbst geeignete Werte.
- Die Teilaufgaben bauen aufeinander auf, d.h. alle Aufgaben zusammen ergeben das gesamte Programm.

(a) Definieren Sie `STOP`, `REORIENT`, `GO`, `BACK`, `TURN`, `AVOID` und `TURNBACK` als Konstanten. Diese entsprechen den zuvor beschriebenen Verhalten des Roboters und müssen in Ihrer folgenden Implementierung als Status (`state`) verwendet werden. Sollten Sie in Ihrer Implementierung weitere Konstanten verwenden, dann definieren Sie diese auch in dieser Teilaufgabe.

(b) Deklarieren Sie die Variablen `state` (`STOP`), `speed` (200) und `obstacle` (1000). Initialisieren Sie die Variablen mit den in Klammern angegebenen Werten. Initialisieren Sie zudem Timer 0 mit dem Wert 0 und die Motorwerte mit 0. Verwenden Sie die Variablen in den folgenden Teilaufgaben und deklarieren Sie hier ggf. zusätzlich genutzte Variablen.

(c) Realisieren Sie nun im Button-Ereignis (`buttons`) den Statuswechsel zwischen Start bzw. Stopp des Roboters. Durch Drücken des vorwärts Knopfes `button.forward` wird der Roboter gestartet und in den `REORIENT`-Status gesetzt. Dies soll jedoch nur möglich sein, wenn er sich im `STOP`-Status befindet. Der `STOP`-Status soll aus jedem beliebigen Status durch ein Drücken des Knopfes in der Mitte `button.center` aktiviert werden können.

(d) Verwenden Sie das Ereignis `prox`, um ein Ausweichmanöver auszulösen, wenn der mittlere horizontale IR-Sensor über dem Schwellenwert `obstacle` liegt. Implementieren Sie hier zusätzlich die Motorsteuerung für die unterschiedlichen Verhalten im Ausweichmanöver. Das Umschalten zwischen diesen geschieht in der nachfolgenden Teilaufgabe.

(e) Timer 0 wird genutzt, um beim Ausweichmanöver zwischen den Verhalten (`BACK`, `TURN`, `AVOID`, `TURNBACK` und `REORIENT`) nach Ablauf der vorgegebenen Zeiten zu wechseln. Setzen Sie dies im Ereignis des Timers 0 mithilfe der Variable `state` nun um.

(f) Das Ereignis `acc` wird verwendet um die Verhalten `REORIENT` und `GO` zu implementieren. Solange die Sensorwerte des Accelerometers auf der y-Achse (`acc[1]`) kleiner als 3 oder auf der x-Achse (`acc[0]`) nicht zwischen -1 und 1 liegen ist der Roboter nicht bergauf gerichtet und soll sich auf der Stelle in eine beliebige Richtung im Kreis drehen. Sobald die Sensorwerte im angegebenen Bereich liegen, soll der Roboter in den Status `GO` wechseln und bergauf fahren. Implementieren Sie hierfür eine Steuerung, bei der die Richtung anpasst wird um die Werte der x-Achse zwischen -1 und 1, also bergauf, zu halten.

Hinweis: Sollte der Wert der x-Achse kleiner als -1 sein, muss der Roboter mehr nach links lenken und dementsprechend bei einem Wert größer als 1 in die entgegengesetzte Richtung.