Mai 2023

# Praktikum zur Lehrveranstaltung Mikrorechentechnik II
# Versuch: ART1 Türsteuerung

Organisation des Praktikumsversuchs: M. Herhold

## Inhaltsverzeichnis



## 1 Versuchsziel

Festigung des Wissens der Vorlesung Mikrorechentechnik 2, durch Anwendung der Sprache C++ mit Fokus auf die objektorientierten Eigenschaften der Programmiersprache. Das Praktikum umfasst den Entwurf und die Implementierung einer Steuerung einer automatischen Tür.

## 2 Einführung

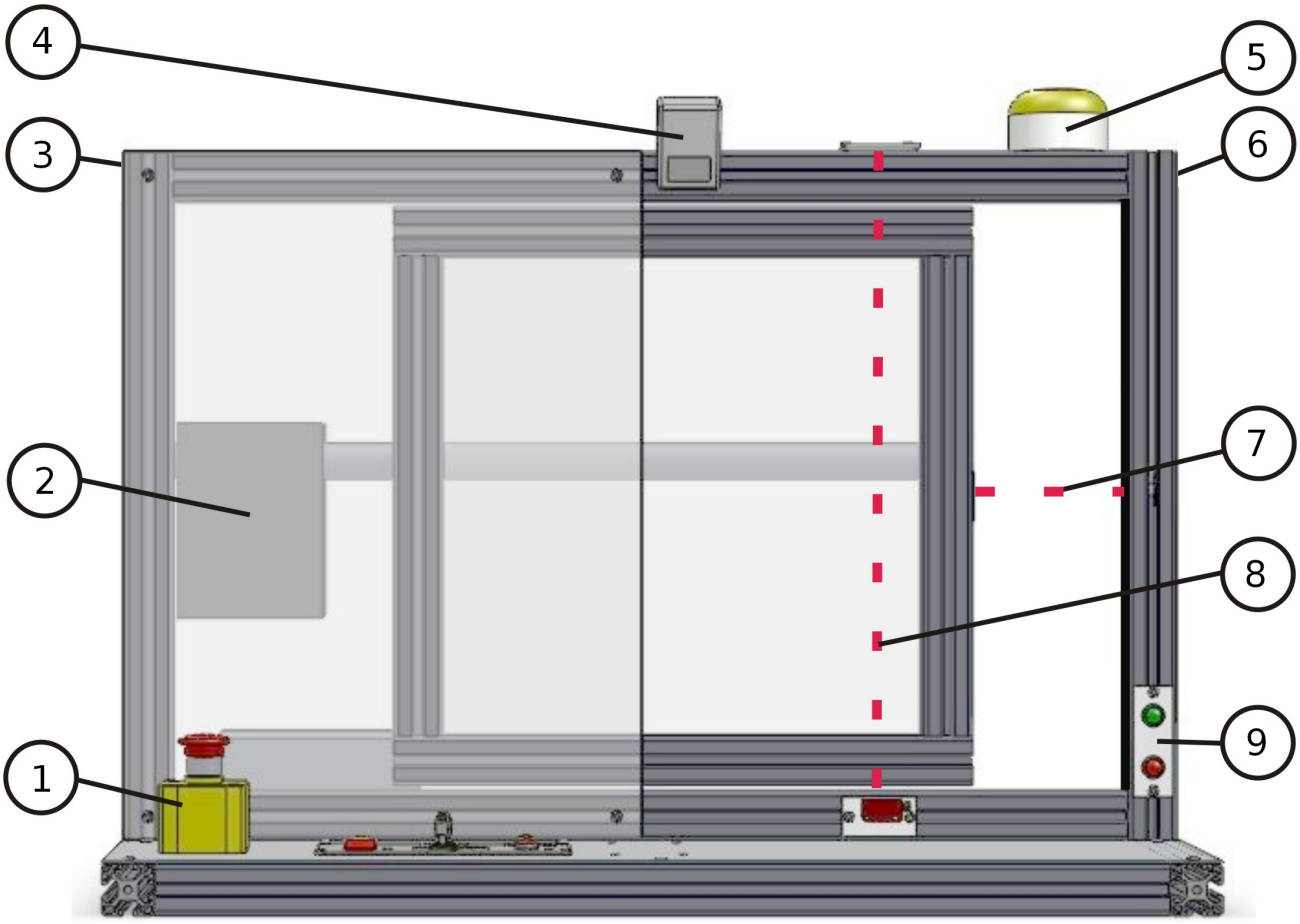
Automatisch bewegte Türen können z.B. bei Fahrzeugen, Garagen, Geschäften, Schutz- und Sicherheitsräumen eingesetzt werden. Automatisierungsziele können u.a. sein, die Passagezeiten zu verringern, den Sicherheitsstandard und einen Komfort nach dem Stand der Technik zu erfüllen. Teilaufgaben für die Automatisierung einer Türsteuerung sind u.a.:

- automatisches Türöffnen bei Nutzeranwesenheit, Befehlseingaben oder im Störfall
- automatisches Türschließen nach Nutzerpassage, Befehlseingabe oder Zeitereignissen
- Verriegelung und Freigabe der Türnutzung
- automatisches Unterbrechen der Türbewegung bei Störungen
- ermöglichen von Reparatur- und Wartungsarbeiten

Im Versuch soll die Steuerungssoftware einer automatischen Tür für verschiedene Einsatzzwecke realisiert werden. Die Steuerung hat die Aufgabe, die Tür bei bestimmten Ereignissen zu öffnen oder zu schließen und soll unterschiedliche in Hardware realisierte Türen steuern. Auch geänderte Sicherheitsvorschriften müssen in der Steuersoftware einfach umsetzbar bleiben. Aus diesen Gründen wird auf die Wart- und Erweiterbarkeit der Software besonders Wert gelegt.

## 3 Beschreibung des gegebenen Versuchsaufbaus

Zum Praktikumstermin steht ein reales Türmodell zur Verfügung. Dieses Türmodell besitzt eine Reihe von Sensoren und Aktoren, welche über eine angeschlossene I/O-Karte ausgelesen bzw. angesteuert werden können. Die I/O-Karte ist auf den Praktikums-PCs betriebsbereit eingebunden. Als Programmier- und Laufzeitumgebung wird die, aus dem Praktikumsversuch 3 – `Animation` des Wintersemesters bekannte, Virtuelle Maschine[1] verwendet. Die technischen Details zur Ansteuerung der Hardware sind im vorgegebenen Programmrahmen bereits fertig implementiert.




Legende Türmodell:

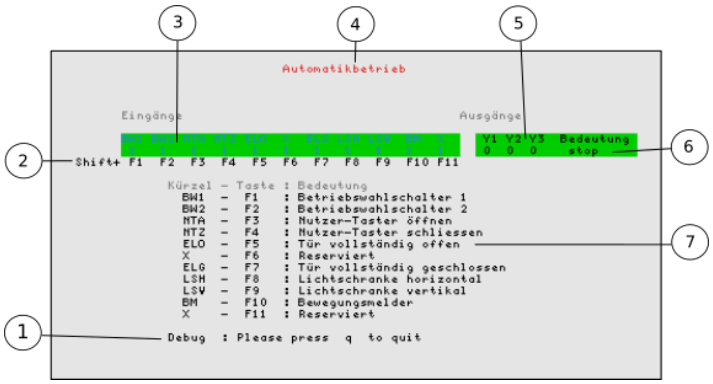
1. Betriebswahlschalter `BW1` und `BW2` *(und Notaus)*
2. Antriebseinheit `Y1` und `Y2`
3. Tür-Endlagesensor Offen `ELO`
4. Bewegungsmelder `BM`
5. Signal-Blinkleuchte `Y3`
6. Tür-Endlagesensor Geschlossen `ELG`
7. Lichtschranke horizontal `LSH`
8. Lichtschranke vertikal `LSV`
9. Nutzer-Taster zum Öffnen `NTA` und Schließen `NTZ`

[1] im nachfolgenden Text nur noch als `VM` bezeichnet

Die Signalleuchte blinkt selbstständig, sobald `Y3` aktiviert ist. Der Notausschalter ist nicht per Software auslesbar.

Das Verhalten der Hardware wird vollständig von der Steuersoftware bestimmt. Somit ist bei fehlerhafter Steuersoftware ein Fehlverhalten der Hardware möglich. Das Türmodell ist so konstruiert, dass die Hardware keinen Schaden nimmt.

Der vorgegebene Programmrahmen simuliert ein technisch eingeschränktes Gerätedisplay.




Legende Gerätedisplay:

1. einzeilige Ausgabemöglichkeit für die Steuersoftware, nutzbar mit `DoorInterface::DebugString()`
2. Tastenkürzel zum Ändern der Sensorzustände im Simulationsmodus
3. Auflistung der Eingänge und deren aktueller Zustand, Kennung und Pegel
4. Anzeige des aktuellen Betriebszustands, abhängig von `BW1` und `BW2`
5. Auflistung der Ausgänge und deren aktueller Zustand
6. im Simulationsmodus Anzeige der Motorbewegung und Signalleuchte
7. Kurzbeschreibung der Eingänge

Der Zustand der Sensor-Eingänge und Aktor-Ausgänge wird durch einen 0- oder 1-Pegel auf dem entsprechenden Port am entsprechenden Pin markiert. `Pin 0` ist nach dem Aufruf von `DoorInterface::DIO_Read()` auf dem niederwertigsten Bit zu finden.

| Port | Pin | Sensor | Aktor | Name |
|---|---|---|---|---|
| 0 | 0 | X | | BW1 |
| 0 | 1 | X | | BW2 |
| 0 | 2 | X | | NTA |
| 0 | 3 | X | | NTZ |
| 0 | 4 | X | | ELO |
| 0 | 5 | X | | reserviert |
| 0 | 6 | X | | ELG |
| 0 | 7 | X | | LSH |
| 1 | 0 | X | | LSV |
| 1 | 1 | X | | BM |
| 1 | 2 | X | | reserviert |
| 1 | 3 | X | | reserviert |
| 1 | 4 | X | | reserviert |
| 1 | 5 | X | | reserviert |
| 1 | 6 | X | | reserviert |
| 1 | 7 | X | | reserviert |
| 2 | 0 | | X | Y1 |
| 2 | 1 | | X | Y2 |
| 2 | 2 | | X | Y3 |
| 2 | 3 | | X | reserviert |
| 2 | 4 | | X | reserviert |
| 2 | 5 | | X | reserviert |
| 2 | 6 | | X | reserviert |
| 2 | 7 | | X | reserviert |

## 4 Aufgabenstellung

Entwerfen, entwickeln und dokumentieren sie im Team eine Steuerungssoftware für eine automatische Tür!
Ihre Software soll ohne Änderungen am Quellcode mit Türen funktionieren, welche zusätzliche oder fehlende Sensoren besitzen!
Das gewünschte Verhalten einer beispielhaften Tür wird im Abschnitt 5 beschrieben.

### 4.1 Entwurf und Design

Verfolgen sie einen **objektorientierten Ansatz** und realisieren sie die einzelnen **Sensoren & Aktoren als Instanzen** von Klassen (mit einer sinnvollen Vererbungshierarchie)!
Erweitern sie die Klasse `DoorControl`. Diese Klasse stellt, durch Ableitung von der Klasse `DoorInterface`, ihre Schnittstelle zur Hardware und Simulationsumgebung dar.
Entwerfen sie eine **zusätzliche Klasse**, welche Details **zur Sensorkonfiguration** aus einer Datei einliest und die Eigenschaft (High- oder Low-aktiv) der einzelnen Sensoren entsprechend verändert.

### 4.2 Entwicklung und Inbetriebnahme

Stellen sie die Steuerung vor dem Praktikumstermin mit Hilfe der Simulation fertig!
Während des Praktikumstermins wird ihre Steuerung am echten Türmodell in Betrieb genommen. Zusätzliche am Modell zu findende Sensoren sollen ihrer Software hinzugefügt werden.
Die Steuerungssoftware muss in der VM kompilieren und lauffähig sein.

### 4.3 Abschließende Dokumentation ihrer Arbeit (Inhalt)

- Einleitung & Ausblick
- Arbeitsweise ihrer Steuerung
- Geben sie die Booleschen Funktionen, die sie zur Steuerung der Tür ausgearbeitet haben in geeigneter Weise an (zB. tabellarisch oder KV-Diagramm oder als Formel).
- Analysieren und dokumentieren sie Probleme welche während der Entwicklung auftraten.
- Diskutieren sie Vor- und Nachteile ihrer Lösung in Hinsicht auf zukünftige Verwendung und Erweiterbarkeit, anhand ihrer Erfahrungen im Praktikumstermin.
- Optional: Dokumentieren sie die Arbeitsweise Ihrer Steuerung mittels eines UML2 Zustandsdiagramms[2].
- Keine seitenweise Quellcode-Listings.

## 5 Beschreibung Türverhalten

**Hinweis:** bei folgender Beschreibung der Tür, handelt es sich um die Aussagen des gedachten zukünftigen Betreiber der Tür. Fehlende, unklare oder widersprüchliche Aussagen sollen durch sinnvolles Verhalten der Software ausgeglichen werden. Für unsinnige Sensorwerte soll selbständig ein geeignetes Türverhalten entworfen und dokumentiert werden. Im Falle einer notwendigen Rücksprache mit dem gedachten zukünftigen Betreiber, nehmen sie bitte Kontakt mit dem Praktikumsbetreuer auf.

### 5.1 Anzahl der Sensoren und Aktoren

Es ist damit zu rechnen, dass zusätzliche noch unbekannte Sensoren oder Aktoren der Hardware (während des Praktikumstermins) hinzugefügt werden, welche Einfluss auf das Verhalten der Tür nehmen sollen. Diesem Umstand ist beim Entwurf der Steuerungssoftware Rechnung zu tragen. Die Software soll ohne Änderungen am Quellcode mit zusätzlichen/fehlenden Sensoren/Aktoren betrieben werden können. (Beispiele: zweiflügelige Tür mit zwei Motoren, Bewegungsmelder oder Taster auf beiden Seiten der Tür, etc.) Das prinzipielle im Folgenden aufgeführte Türverhalten bleibt dabei bestehen.

### 5.2 Interpretation der Sensorzustandswerte

Normalerweise sind die Sensoren und Aktoren High-aktiv (`1-Pegel` bedeutet aktiv), es gibt aber auch Hardwaremodelle der Tür bei denen einzelne oder alle Sensoren Low-Aktiv (`0-Pegel` bedeutet aktiv) sind. Die Steuersoftware soll diesem Umstand Rechnung tragen, indem sie sich dieses Detail für jeden Sensor per Konfigurationsdatei konfigurieren lässt. Die Konfigurationsdatei soll beim Programmstart einmalig eingelesen und beachtet werden. Die konkrete Strukturierung und Inhalt der Konfigurationsdatei ist dem Entwicklerteam überlassen.

[2] Erstellung von UML-Diagrammen mit PlantUML: https://plantuml.com/de/state-diagram

## 5.3 Aktorzustandswerte

sind bei jedem Modell High-aktiv.

## 5.4 Grundsätzliches Verhalten der Steuerungssoftware

- Sobald die Tür eine Endlage (komplett offen oder geschlossen) erreicht hat, soll der Motor abgeschaltet werden und die Signalleuchte soll aufhören zu blinken.
- Die Ausgänge `Y1` und `Y2` gleichzeitig aktiv zu schalten, ist ein ungültiger Zustand und soll unbedingt vermieden werden.

## 5.5 Betriebsarten:

Es gibt 4 mögliche Betriebsarten welche durch `BW1` und `BW2` geschaltet werden:

- Automatikbetrieb, Handbetrieb, Reparaturbetrieb, Ausgeschaltet

### 5.5.1 Automatikbetrieb

Im Automatikbetrieb befinden sich beide Betriebswahlschalter `BW1` und `BW2` im `Aktiven`-Zustand.
Aus Sicherheitsgründen soll die Signalleuchte 5 Sekunden nach dem Aktivieren der Tür leuchten um anzuzeigen, dass die Automatik aktiv ist. Während dieser Zeit, soll keine Aktion seitens der Tür erfolgen, alle Eingaben und Sensorwerte werden verworfen. Anschließend arbeitet die Automatiksteuerung nach folgendem Schema:

1. Sobald zu einem beliebigen Zeitpunkt an einem oder mehreren Sensoren `NTA`, `LSH`, `LSV` oder `BM` ein aktiver Zustand detektiert wurde, öffnet sich die Tür bis diese den vollständig geöffneten Zustand erreicht hat.
2. Nachdem die Tür vollständig geöffnet wurde, verharrt diese für 3 Sekunden in dieser Position, bevor sich die Tür selbständig schließt. Aktive Zustände an den Sensoren `NTA`, `LSH`, `LSV` oder `BM` starten diesen 3 Sekunden Timer sofort von vorn.
3. Ein Betätigen des Tasters `NTZ`, zu einem beliebigem Zeitpunkt, soll zum augenblicklichen Schließen der Tür führen. Die Tür soll sich unabhängig davon, ob `NTZ` gedrückt bleibt oder nicht, vollständig schließen. Automatikbetrieb-Regel 1 ist zu beachten.
4. Denkbar wäre ein zusätzlicher, noch nicht vorhandener, Sicherheitssensor, welcher die Tür beim Schließen sofort stoppt.

### 5.5.2 Handbetrieb:

Dieser Betriebsmodus ist dadurch definiert, dass `BW1` aktiv und `BW2` inaktiv ist.

1. Die Tür öffnet sich vollständig bis zur Endlage, sobald `NTA` betätigt wurde.
2. Die Sensoren `LSH`, `LSV` und `NTZ` werden ignoriert, solange die Tür sich öffnet.
3. Die Tür schließt sich vollständig bis zur Endlage, sobald `NTZ` betätigt wurde.
4. Einklemmschutz: mindestens ein aktiver Sensor `LSH`, `LSV` oder `NTA`, stoppt die Tür sofort, falls diese sich gerade schließt.
5. Die Sensoren `LSH` und `LSV` werden ignoriert, falls die Tür sich nicht bewegt.
6. Der Sensor `BM` und die Signalleuchte `Y3` haben keine Funktion.

### 5.5.3 Reparaturbetrieb

In diesem Betriebsmodus ist `BW1` inaktiv und `BW2` aktiv.

1. Die Tür bewegt sich nur, solange exakt ein Sensor (`NTA`, `NTZ`, `LSH`, `LSV` oder `BW`) aktiv ist. `NTZ` schließt die Tür, die restlichen Sensoren öffnen die Tür.
2. Sind keine oder mehr als einer der oben genannten Sensoren aktiv, bleibt die Tür sofort stehen.
3. Die Signalleuchte soll aktiviert sein, solange die Tür sich bewegt.

#### 5.5.4 Ausgeschaltet

Wenn `BW1` und `BW2` beide inaktiv sind soll die Anlage den Zustand abgeschaltet haben.

1. Die Tür reagiert auf keine Sensoren, außer auf `BW1` und `BW2`.

## 6 Vorschläge zu Design und Entwicklung der Steuerung (optional)

- Es ist sinnvoll die Steuerung als Zustandsautomaten zu entwerfen.
- Überlegen sie sich gemeinsam im Team, welche Klassen und welche Hierarchie sinnvoll ist. Verteilen sie die so entstandenen Klassen auf Teammitglieder und spielen sie in einer Art Rollenspiel das Zusammenwirken der Klassen während des Programmablaufs durch.
- Kategorisieren (`Öffnen`, `Schließen`, `Sicherheit`, ...) sie die Sensoren und verwenden sie Container (zB. `std::list`) und Algorithmen (zB. `std::any_of`) welche C++ anbietet, um eine variierende Anzahl aktiver Sensoren zu beherrschen.
- Die Instanz eines Sensors kann intelligent mit Hilfe von `get()`-/ `set()`-Methoden entscheiden, ob der Sensor aktiv ist. Durch diese Herangehensweise ist es einfach auf die unterschiedliche Interpretation (High-aktiv, Low-aktiv) der Sensorwerte zu reagieren. (Stichwort `Datenkapselung`)
- Die Klasse, welche die Konfigurationsdatei verarbeitet, könnte die Instanzen der Sensoren mit der entsprechenden Konfiguration erzeugen.
- Vorteilhaft ist ein menschenlesbares Format für die Konfigurationsdatei um Änderungen und Prüfungen an der Konfiguration mit einem Texteditor vornehmen zu können.
- Aufgrund der (absichtlich) eingeschränkten Fähigkeiten des (simulierten) Displays könnte es angebracht sein, Zustands- und Debug-Informationen, zum Zweck der Fehleranalyse, in einer Datei zu loggen. Ausgaben mittels `printf()` oder `std::cout «` stören das simulierte Display.
- Halten sie einzelne Funktionen/Methoden kurz und übersichtlich. Teilen sie lange komplizierte Funktionen/Methoden besser auf mehrere Funktionen/Methoden auf. Funktionen/Methoden deren Quellcode nicht ohne zu scrollen auf dem Bildschirm angezeigt werden können, sind zu lang und sollten aufgeteilt werden.
- Es ist ihnen freigestellt, welche Entwicklungswerkzeuge sie verwenden. Wichtig ist, dass ihre Steuersoftware in der VM kompiliert und läuft.

### 6.1 Hinweise zur Inbetriebnahme des vorbereiteten Programmrahmens

Bei dem vorbereiteten Programmrahmen handelt es sich um einen kompletten `Eclipse-Workspace`, welcher, nach dem Entpacken, über folgenden Menüpunkt in Eclipse geladen werden kann:

```
"File" → "Switch Workspace" → "Other" ... (dann Auswahl des Workspace-Verzeichnisses und Bestätigen mit "OK")
```

Nach erfolgreichem Kompilieren in Eclipse oder per `Makefile`, können sie das Programm starten. Das fertig kompilierte Programm muss in einem laufenden `Terminal`-Programm (zB. xterm) gestartet werden. Der Eclipse-Menüpunkt `External Tools → run in xterm` hilft ihnen dabei.

Nach dem Start des Programms sehen sie das simulierte Gerätedisplay.

#### 6.1.1 FAQ:

- *Wie bekomme ich Dateien in die VM?*: Sie haben mehrere Möglichkeiten Dateien mit der VM auszutauschen. Über die Einstellungen der VM, können sie vor dem Start der VM einen `Gemeinsamen Ordner` festlegen, welcher dann auf ihrem PC und in der VM sichtbar ist. Sie können auch einen physischen USB-Massenspeicher über das `Geräte`-Menü in die VM einbinden. Sie können mit `git` arbeiten und ein selbst angelegtes Repository klonen. Sie können die Daten mittels `wget` von einem Web-Server herunterladen.
- *Warum kann ich mein Programm in Eclipse in der VM nicht debuggen?*: Installieren sie den GNU-Debugger indem sie das Paket `gdb` in der VM installieren.
  Shell Kommando: `sudo apt-get update; sudo apt-get install gdb`

# 7 Hinweise zum Praktikumstermin und dessen Durchführung

## 7.1 Vorbereitung

- Falls sie bisher keine/wenig Erfahrung mit C++ gemacht haben, planen Sie mehre Tage/Wochen zum Erlernen der notwendigen Kenntnisse und die Fehlersuche ein.
- Arbeiten sie die Aufgabenstellung und die Arbeitssicherheitshinweise in diesem Dokument durch.
- Laden sie den vorbereiteten Programmrahmen (Eclipse Workspace) herunter. (siehe Abschnitt Downloads)
- Stellen sie ihre Steuerungssoftware zu Hause, mit Hilfe der VM (siehe Downloads), soweit wie möglich fertig.
- Bringen sie den von Ihnen so vorbereiteten Eclipse Workspace zur Versuchsdurchführung mit. (*Hinweis:* Archivierungsdateiformat nutzen, z.B.: `zip` oder `tar.gz` um die Datenintegrität zu gewährleisten)
- Fragen zur Versuchsaufgabe und Programmierung oder bei Problemen bezüglich der Lösung der Aufgabe können sie per E-Mail an M.Herhold[3] (Lehrstuhl für Automatisierungstechnik) stellen. Klar formulierte Fragen führen nicht zu Punktabzug bei der Leistungsbewertung. Die Beantwortung erfolgt während der regulären Büroarbeitszeiten.

## 7.2 Aufbau und Nutzung des Versuchsplatzes

Der Versuchsplatz ist ein PC mit Linux und vorhandener Internetanbindung. Auf dem System ist die VM installiert und betriebsbereit. Das Türmodell ist angeschlossen und betriebsbereit. Ihr Quellcode muss zum Bestehen des Versuchs in der VM kompilieren und das Programm in der VM laufen.

Die Übertragung ihres Quellcodes ist per USB-Stick oder Download aus dem Internet möglich.

## 7.3 Aufgaben während der praktischen Versuchsdurchführung zum Praktikumstermin

Passen sie ihre vorbereitete, fertige und in der Simulation erprobte Steuerungssoftware an die reale Hardware und deren abweichendes Sensorverhalten an. Erproben sie systematisch die Funktion ihrer Steuerungssoftware am Türmodell.

Präsentieren sie ihr Anwendungsprogramm dem anwesenden Betreuer und beantworten sie Fragen zu ihrem Quellcode. Nach erfolgreicher Präsentation ihrer Anwendung, kann der Versuchstermin vorzeitig beendet werden. Die Präsentation geht in die Bewertung des Praktikumsversuchs ein.

Der so entstandene Quellcode verbleibt in Kopie, für Archivierungs- und Bewertungszwecke, beim Praktikumsbetreuer.

### 7.3.1 Kontrollfragen zur Präsentation

- Begründen sie den Entwurf ihrer programmiertechnischen Lösung.
- Begründen und erklären sie Implementierungsdetails.
- Erklären sie ihre Vorgehensweise zum Test der korrekten Implementierung.

## 7.4 Dokumentation

Zum Bestehen des Praktikums ist von jeder Praktikumsgruppe eine gemeinsame Dokumentation, innerhalb von 1 Woche nach dem Versuchstermin, abzugeben.

Inhalt der Dokumentation, siehe Aufgabenstellung im Abschnitt 4.3.

Die Dokumentation ist zusammen mit dem zu verwendenden Deckblatt, per E-Mail an M.Herhold[3], im PDF-Format einzureichen.

Die Dokumentation fließt in die Bewertung des Praktikumsversuchs ein.

[3] mario.herhold `ätt` tu-dresden.de

## 8 Downloads

**Virtuelle Maschine, Programmrahmen (Eclipse Workspace) und Dokumentationsdeckblatt** siehe Webseite des Lehrstuhls für Automatisierungstechnik zum Praktikum Mikrorechentechnik 2, ART-1 und ART-3

## 9 Programmierschnittstelle der `DoorInterface`-Klasse

```
/*
 * Boolsche Variable zum Detektieren des gewuenschten Programmabbruchs
 * ‘true‘ bedeutet der Programmabbruch ist gewuenscht
 * den Variablenwert regelmaessig pruefen! */
static std::atomic<bool> quit_doorcontrol_flag;

/*
 * Konstruktor des Interface zur Hardware / Simulation
 * real_door: legt fest ob auf die Hardware zugegriffen werden soll
 * show_ui:   legt fest, ob die Ein-/Ausgabe ueber ncurses genutzt wird
 * wirft Exception ‘std::runtime_error‘ im Fehlerfall */
DoorInterface(bool real_door=false, bool show_ui=true);

/*
 * Liest alle Pins eines Eingabe-Ports.
 * port: Port-Nummer
 * bits: ausgelesenen Kanaele des Ports (Rueckgabewert) */*
void DIO_Read(const unsigned port, unsigned char *pins);

/*
 * Aendert alle Pins eines Ausgabe-Ports.
 * port: Ausgabe-Port-Nummer
 * pins: die auszugebenden Kanaele des Ports */
void DIO_Write(const unsigned port, const unsigned char pins);

/*
 * Methode zum Anzeigen eines Strings in der Nachrichtenzeile des Displays
 * s: auszugebender String (maximale Laenge 50 alphanumerische Zeichen) */
void DebugString(const std::string s);
```

## 10 Arbeits- und Brandschutzhinweise

### 10.1 Vorbeugende Maßnahmen:

- Die Praktikumsteilnehmer haben sich so zu verhalten, dass Gefahrensituationen und Unfälle vermieden werden.
- Die Befugnis zum Bedienen und Nutzen von Geräten ist auf den zugewiesenen Praktikumsplatz beschränkt.
- Eingriffe in die zum Praktikumsaufbau gehörenden Geräte sind nicht erlaubt.
- Der Anschluss und der Betrieb privater Geräte in den Praktikumsräumen ist verboten.
- Defekte an Geräten oder Gebäudeeinrichtungen sind unverzüglich dem Betreuer mitzuteilen. Betroffene Geräte sind außer Betrieb zu nehmen. Andere Personen sind vor Gefahren zu warnen.
- Den Anweisungen der Praktikumsbetreuer bzw. anderer aufsichtsführender Personen ist unbedingt Folge zu leisten.
- Betriebsfremde dürfen sich nur mit Erlaubnis des Praktikumsbetreuers in den Praktikumsräumen aufhalten.
- Rauchen und Umgang mit offenem Feuer ist nicht gestattet.
- Nach Ende des Praktikums ist der Arbeitsplatz in sauberem und aufgeräumtem Zustand zu hinterlassen.
- Außergewöhnliche Ereignisse bzw. besondere Vorkommnisse sind umgehend dem Betreuer oder dem diensthabenden Assistenten zu melden.

### 10.2 Verhalten im Falle eines Brandes:

- Beachten der richtigen Reihenfolge: **MELDEN - RETTEN - LÖSCHEN**

#### 10.2.1 Feuer melden:

- Telefonische Brandmeldung:
  - Notruf 112 der Feuerwehr (von jedem Telefon aus möglich)
  - Notruf HA 34515 der Technischen Leitzentrale der TUD
- Deutliche, genaue und vollständige Angaben:
- Wo brennt es?
- Was brennt?
- Angaben zu verletzten oder gefährdeten Personen
- Wer meldet?

#### 10.2.2 Personen retten:

- Erste Hilfe leisten
- Weitere Hilfe organisieren, medizinische Hilfe anfordern
- Gefahrenbereich räumen; Fluchtwege benutzen, keine Aufzüge
- Andere Personen warnen, Sammelplatz (Platz vor Turmeingang zum Barkhausenbau) aufsuchen
- Behinderten und älteren Personen helfen

#### 10.2.3 Löschversuch unternehmen

- Feuerlöscher verwenden (Standorte: Gänge des Barkhausenbaues), dabei sich nicht selbst gefährden
- Fenster und Türen schließen, aber nicht abschließen
- Möglichst elektrische Verbraucher abschalten

### 10.3 Rufnummern für Notfälle:

| Helfer | Telefonnummer |
|---|---:|
| Rettungsdienst | 112 |
| Polizei | 110 |
| TUD-Notruf | 34515 |
| Betriebsärztlicher Dienst | 36199 |
| Klinikum Friedrichstadt Notaufnahme | 0351-480 1938 |