­­­

**TGM Wien**

**DezSys 07 – Pi Calculator**

**SYT 2014/2015 | Stand: 12.12.2014**

**Matthias El-Far & Michael Weinberger 4AHITT**

Inhalt

[Aufgabenstellung 2](#_Toc406017606)

[Beschreibung auf Moodle 2](#_Toc406017607)

[Designüberlegung 3](#_Toc406017608)

[Erster Ansatz 3](#_Toc406017609)

[Konkrete Idee 3](#_Toc406017610)

[Umsetzung 3](#_Toc406017611)

[Detaillierte Arbeitsaufteilung (Aufwandsabschätzung, Endzeitaufteilung) 4](#_Toc406017612)

[Aufgabentrennung 4](#_Toc406017613)

[Aufwandabschätzung 4](#_Toc406017614)

[Endzeitaufteilung 4](#_Toc406017615)

[Fazit 4](#_Toc406017616)

[Arbeitsdurchführung 5](#_Toc406017617)

[Resultate 5](#_Toc406017618)

[Niederlagen 5](#_Toc406017619)

[*Quellenangaben:* 6](#_Toc406017620)

# Aufgabenstellung

## Beschreibung auf Moodle

Aktivieren Sie den Pin PA0 auf PortA um den User-Button verwenden zu können. Beachten Sie, dass dieser Pin als Input-Pin konfiguriert werden muss (Pull-Up/Down nicht vergessen!). Schreiben Sie Ihren Code der Aufgabe IndInf06 um und reagieren Sie folgendermaßen auf das Drücken des Buttons:

Button != gedrückt -> Lichter° %90° == 0 -> Licht an, Andere aus

Button == gedrückt -> Lichter° %90° == 0 -> Licht aus, Andere an

**Implementieren Sie auch eine Funktion zum Toggeln der LEDs!**

*Dokumentieren Sie alle notwendigen Schritte und Einstellungen und schreiben Sie ein Protokoll.*

*Dies ist keine Gruppenarbeit!*

# Designüberlegung

## Erster Ansatz

Ich nehme mir die Aufgabe IndInf06 her und baue eine If-Anweisung ein, die mir sagt, wann der Knopf gedrückt ist und wenn nicht, was dann passiert.

## Konkrete Idee

Im Ruhezustand sollen die LEDs wie ein Plus leuchten, solange der User-Knopf gedrückt ist sollen die angeschalteten LEDs ein Kreuz bilden.

Es wird eine Methode benötigt, mit der LEDs an- und abgeschaltet (getogglet) werden können.

## Umsetzung

Auf Basis der vorherigen Übungen, habe ich mich im Datenblatt informiert, wie der Button ‚ausgelesen‘ wird.

|  |
| --- |
| init\_clock**();**    // Port A & E verwenden  GPIO\_TypeDef **\***PE **=** GPIOE**;**  GPIO\_TypeDef **\***PA **=** GPIOA**;** |

Hier habe ich die obere Zeile kopiert und logisch daraus geschlossen, dass ich so auch den *GPIOA*-Port verwenden kann. Dies erwies sich nach einem Blick ins Datenblatt auch als richtig.

|  |
| --- |
| // Mode auf Output setzen  PE**->**MODER **=** **(**1 **<<** **(**15**\***2**)** **|** 1 **<<** **(**14**\***2**)** **|** 1 **<<** **(**13**\***2**)** **|** 1 **<<** **(**12**\***2**)** **|** 1 **<<** **(**11**\***2**)** **|** 1 **<<** **(**10**\***2**)** **|** 1 **<<** **(**9**\***2**)** **|** 1 **<<** **(**8**\***2**));** |

Wieder werden alle LEDs auf *Output* gesetzt.

# Detaillierte Arbeitsaufteilung (Aufwandsabschätzung, Endzeitaufteilung)

## Aufgabentrennung

Einzelarbeit, obsolet

## Aufwandabschätzung

Dieses Projekt sollte in 3-4 Stunden fertiggestellt werden.

## Endzeitaufteilung

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Planung/  Verstehen | Reinlesen in OO PHP | Statische Implementie-rung | Implementierung dynamisch mit Parametern | Datenbank-Zugriff | Debug-gen | Protokoll | End-ergebnis |
| 0,5 | 0,25 | 2 | 1 | 1 | 0,5 | 0,75 | 6 h |

Endaufwand: *6 Arbeitsstunden*, verteilt auf 3 Tage.

## Fazit

# Arbeitsdurchführung

## Resultate

## Niederlagen

# *Quellenangaben:*

*http://php.net/.... (einiges durchforstet)*

*Head First Design Patterns 🡪 SÜ*