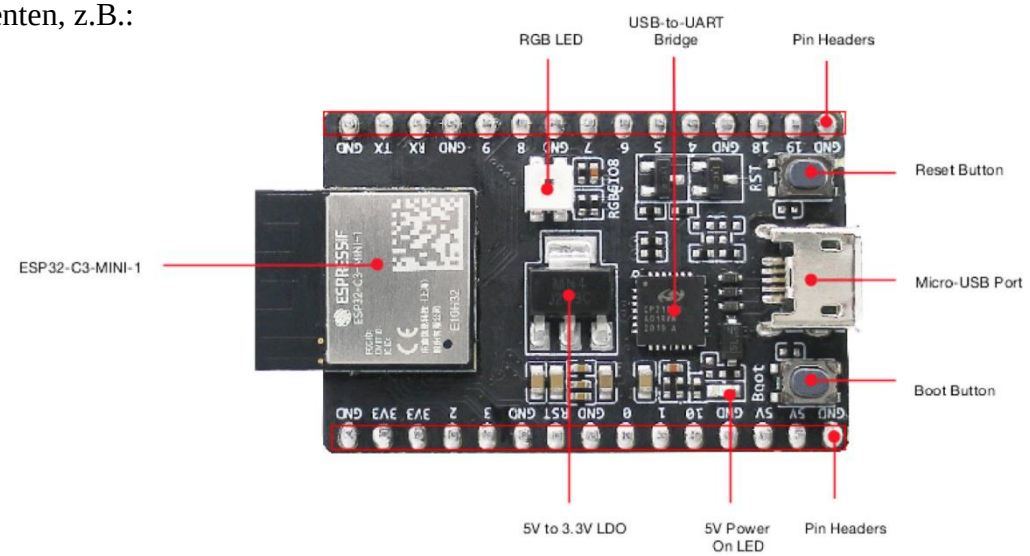
Systemnahe Programmierung 2

DHBW Stuttgart, 2024
Prof. Dr.-Ing. Alfred Strey

Laborübung 1

Die vorliegende Platine DevKitM-1 mit dem ESP32-C3 Mikrocontroller verfügt über diverse Komponenten, z.B.:



- ESP32-C3 Mikrocontroller mit WiFi-Antenne
- Spannungsregler (LDO)
- Reset-Button `S1`, verbunden mit CHIP_PU, lässt ein geladenes Programm neu starten
- Boot-Button `S2`, verbunden mit GPIO 9, wird hier nicht benötigt!
- `Power-LED` (rot), über Widerstand fest verbunden mit 5 V
- `RGB-LED`, verbunden mit GPIO 8
- Zwei 15-polige Steckerleisten `J1` und `J2` mit Signalen diverser E/A-Pins des ESP32-C3 sowie Anschlüssen für 3.3V, 5V und GND

Das Board arbeitet mit einer Spannung von 3,3V, die über einen Spannungsregler aus den 5V des USB-Anschlusses erzeugt werden.

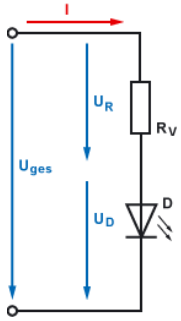
Aufgabe 1: Erste Schritte mit Visual Studio Code und ESP32-C3

1) Schalten Sie den Laborrechner ein und loggen Sie sich mit dem Account "student" ein; das Passwort wird Ihnen beim Laborversuch mitgeteilt.

2) Machen Sie sich mit dem Schaltplan des ESP32-C3 Boards und der Kurzdokumentation vertraut.

3) Stecken Sie das ESP32-C3 Board am Rand des Experimentierbretts ein, mit dem Micro-USB Port nach außen zeigend.

4) Starten Sie *Visual Studio Code*, z.B. über die Windows-Taste und folgender Eingabe des Namens in der Suchmaske. Es ist sinnvoll, *Visual Studio Code* als Icon in der Menuleiste am linken Bildschirmrand hinzuzufügen.

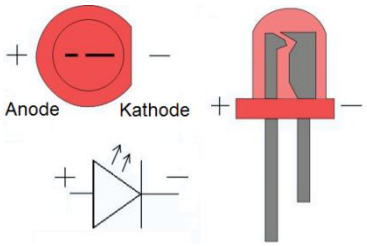


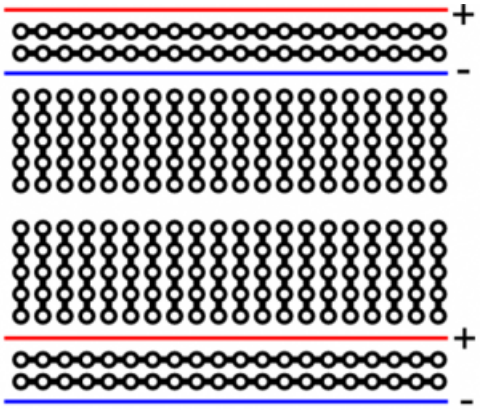
5) Schließen Sie extern an den Pin GPIO 2 (zugehörige Platinenbeschriftung des Anschlusses ist "2") über einen Vorwiderstand eine LED an. Berechnen Sie zunächst den erforderlichen Widerstand, wenn an der LED eine Spannung von U_D = 2,1 V abfällt und der Strom I = 10 mA nicht überschritten werden sollte. Die I/O-Pins des ESP32-C3 erzeugen eine Spannung von 3.3V (für logisch 1)

Hinweise:

– Bitte beachten Sie die interne Verdrahtung des Experimentierboards (s. Abb.). Die Masseleitung (–) ist mit einem GND-Pin von Steckerleiste `J1` oder `J2` des Boards zu verbinden.

– Achten Sie auf die korrekte Polarität der LED:





6) Verbinden Sie das Board (am Micro-USB Port) über das USB-Kabel mit einer USB-A Buchse des Rechners.

7) Beginnen Sie durch Auswahl von *View → Command Palette → ESP-IDF: New Project* ein neues Projekt, wählen Sie in dem folgend erscheinenden Menu

– einen *Project Name* (beliebiger String ohne Leerzeichen)
– ein existierendes *Project Directory* (z.B. `/home/student/labor`, zuvor anlegen!), in dem das Projekt gespeichert wird
– das ESP-IDF Board *ESP32-C3 chip* (*via builtin USB-JTAG*)

Den *Serial Port* (USB-Port) kann man später noch festlegen! Klicken Sie auf *Choose Template*, und wählen Sie im dann erscheinenden Menu ESP-IDF (anstatt *Extension*) und dann das *sample_project* aus. Klicken Sie abschließend auf den Button *Create project using template sample_project*.

8) Wählen Sie dann *File → Open Folder → Project Directory / Project Name* um das in Schritt 7) angelegte Projekt zu öffnen.

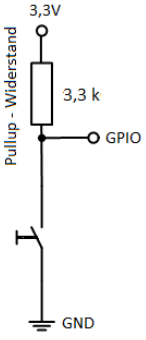
9) Fügen Sie die Dateien `main.c` und `esp32-c3.h` aus dem Verzeichnis `Labor1` der Moodle-Seite unter *main* dem Projekt hinzu.

10) Korrigieren Sie in der Menuleiste am unteren Bildschirmrand ggf. die Prozessorversion in *ESP32-C3* und wählen Sie unter *Select Port to Use* als Serial Port denjenigen USB-Port (z.B. /dev/ttyUSB0) aus, bei dem das ESP32-C3 Board angezeigt wird.

10) Compilieren Sie Ihr Programm über *ESP-IDF: Build Project* z.B. in der Menuleiste am unteren Bildschirmrand oder über *View → Command Palette → ESP-IDF: Build your project.* Nur beim ersten Übersetzen werden auch alle über 900 Quelltextdateien des Laufzeitsystems übersetzt.

12) Wählen Sie über *ESP-IDF: Select Flash Method* in der Menuleiste am unteren Bildschirmrand die Variante UART und klicken Sie dann in der gleichen Leiste auf *ESP-IDF: Flash Device.* Das Programm startet dann nach dem Herunterladen automatisch und lässt die externe LED blinken.

Aufgabe 2: Externe LED ein- und ausschalten



Nun soll die an einen digitalen I/O-Port angeschlossene externe LED über einen Taster eingeschaltet und durch nochmaliges Betätigen des Tasters wieder ausgeschaltet werden.

1) Erarbeiten Sie im *Technical Reference Manual* die Grundlagen der Programmierung der Register für die GPIO Ports.

2) Schließen Sie einen externen Taster gemäß der Skizze rechts über einen Pullupwiderstand an den GPIO Pin 7 an; die externe LED soll über einen Vorwiderstand am GPIO Pin 2 angeschlossen sein.

3) Schreiben Sie nun ein kleines C-Programm, das direkt die Register `GPIO_ENABLE_REG`, `GPIO_OUT_REG`, `GPIO_IN_REG` und ggf. `IO_MUX_GPIO7_REG` anspricht, um in einer Endlosschleife den externen Taster abzufragen und die externe LED ein-/auszuschalten. Bitte beachten Sie, dass Sie nur die jeweils benötigten Bitleitungen des GPIO-Ports und des IO_MUX verwenden und die anderen Bitleitungen unverändert lassen!

Hinweise:

- Auch bei längerem Drücken des Tasters soll es nur eine Zustandsänderung geben!
- Sie finden die Zuordnung der Register zu den jeweiligen Adressen im Adressraum des ESP32-C3 übrigens in der über `#include <esp32c3_reg.h>` inkludierten Datei

Aufgabe 3: Blinken mit Timer

Nun soll ein Timer des ESP32-C3 eingesetzt werden, um die externe LED mit einer Frequenz von 1 Hz (d.h. 1/2 Sekunde an, 1/2 Sekunde aus, …) blinken zu lassen, nachdem diese über den externen Taster eingeschaltet wurde.

1) Machen Sie sich im Kapitel 11 (Timer Group) des *ESP32-C3 Technical Reference Manual* mit der Programmierung des Timers des ESP32-C3 vertraut.

2) Berechnen Sie den benötigten Wert `TIMER_T0_DIVIDER` des Prescalers für das Taktsignal des Zählers und den Wert für die Zeitdauer `TIMG_T0_ALARMLO_REG` des Timers `T0`, nach deren Ablauf ein Alarm ausgelöst werden soll. Der Zähler soll von 0 bis `TIMG_T0_ALARMLO_REG` aufwärts zählen; hierzu ist das Bit `TIMG_T0_INCREASE` entsprechend zu setzen. Nach Abschluss der Konfiguration ist der Zähler über `TIMG_T0_EN` zu aktivieren.

3) Modifizieren Sie Ihr C Programm aus Aufgabe 2, damit die LED im eingeschalteten Zustand blinkt. Überprüfen Sie hierzu durch wiederholtes Abfragen des Bits `TIMG_T0_ALARM_EN` im Register `TIMG_T0_CONFIG_REG`, ob die Zählperiode abgelaufen ist und damit der Zustand der LED gewechselt werden soll.

Hinweis: Das Kontrollbit `TIMG_T0_ALARM_EN` ist nach Erreichen des Alarmwertes erneut zu setzen!

4) Überprüfen Sie mit dem Sekundenzeiger einer Uhr, ob die LED ein Mal pro Sekunde aufblinkt; passen Sie ggf. die Periodendauer des Zählers an.

Aufgabe 4: LED Lauflicht und Debugger

Es soll nun mit 5 LEDs ein Lauflicht implementiert werden, das von links (nur erste LED leuchtet) nach rechts (nur fünfte LED leuchtet) und dann wieder zurück läuft.

1) Schließen Sie hierzu fünf LEDs mit den Vorwiderständen an 5 logisch benachbarte (!) und auf dem Board ungenutzte GPIO Pins über die Steckerleisten `J1` und `J2` an.

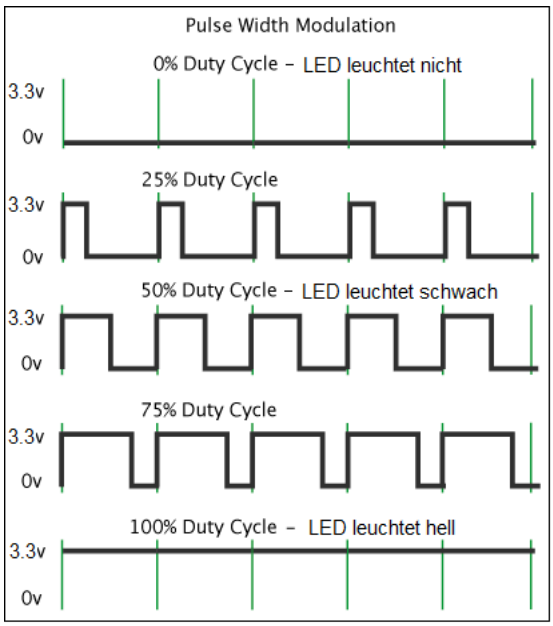
2) Schreiben Sie ein C-Programm, das durch Programmierung der GPIO-Register das Lauflicht erzeugt. Das Lauflicht soll für jeden Durchlauf in einer Richtung 1 Sekunde benötigen. Verwenden Sie hierzu wie in Aufgabe 3 den Timer.

3) Machen Sie sich mit den Debug-Möglichkeiten des ESP32-C3 vertraut. Um über USB debuggen zu können, ist ein spezielles Kabel nötig, das auf der einen Seite einen USB-A Stecker, auf der anderen Seite 2 kleine Doppelstecker hat. Stecken Sie den einen Doppelstecker mit dem weißen Kabel an Anschluss 18 (USB_D–) und mit dem grünen Kabel an Anschluss 19 (USB_D+) des ESP32-C3 Boards in das Experimentierbrett. Der zweite Doppelstecker ist mit dem roten Kabel gegenüber an 5V und dem schwarzen Kabel am benachbarten Pin GND anzuschließen.

Setzen Sie nun einen Breakpoint (durch Mausklick auf die Zeilennummer des Quelltextes), übersetzen und starten Sie Ihr Programm über den Button *Debug Project* in der Menuleiste am unteren Bildschirmrand, Schauen Sie sich am Breakpoint die Inhalte von Variablen an und gehen Sie in Einzelschritten durch Ihr Programm.

Aufgabe 5: Fading einer LED mit PWM

Man kann die Helligkeit einer LED steuern, indem man sie durch ein PWM-Signal ansteuert und das Tastverhältnis ändert (s. Abb. rechts).



1) Machen Sie sich im Kap. 32 des *Technical Reference Manuals* mit der Programmierung der Register der PWM-Einheit vertraut.

2) Die PWM-Einheit ist zur Einsparung von Energie nach dem Einschalten deaktiviert und muss zunächst durch Setzen von `SYSTEM_LEDC_CLK_EN` und anschließendem Reset mittels `SYSTEM_LEDC_RST` aktiviert werden (vgl. Kap. 16.3.5 des *Technical Reference Manuals*).

3) Wählen einen Pin GPIO x, um hier über einen Vorwiderstand eine LED anzuschließen. Aktivieren den Ausgang über `GPIO_ENABLE_REG` und setzen Sie den Wert `GPIO_FUNCx_OUT_SEL` bei Verwendung von PWM-Kanal 0 auf die Nr. des Signals `ledc_ls_sig_out0`.

4) Programmieren Sie den Timer 0 der PWM-Einheit derart, dass der Zähler mit einer Frequenz von 1 MHz von 0 bis 1023 zählt, die Frequenz des PWM-Signals somit ungefähr 1 kHz beträgt. Hierzu sind der Frequenzteiler `LEDC_CLK_DIV_TIMER0` und die Periodendauer `LEDC_TIMERx_DUTY_RES` zu konfigurieren.

5) Programmieren Sie den PVM-Kanal 0 derart, dass der *hpoint* (in `LEDC_CH0_HPOINT_REG`) auf 0 bleibt, der *lpoint* (in `LEDC_CH0_DUTY_REG`) hingegen zwischen 0 und 1023 variiert wird und somit ein Tastverhältnis zwischen 0% und 100% erzielt wird.

6) Schreiben Sie nun ein Programm, das ein "Fading" der LED realisiert, d.h. die Helligkeit der LED soll in einer Endlosschleife langsam und kontinuierlich von einem Minimum (Schwaches Glimmen der LED) zu einem Maximum (volle Helligkeit) ansteigen und dann wieder langsam vom Maximum zum Minimum abnehmen.

Hinweise:

- Die Verwendung von Interrupts ist hier nicht erforderlich!
- Die nötigen Verzögerungen können entweder wie in Aufgaben 3 und 4 mit einem Timer oder mit einer `for`-Schleife realisiert werden.

7) Schauen Sie sich das generierte PWM-Signal auf dem Oszilloskop an.

Wichtiger Hinweis am Schluss: Sichern Sie alle Quelldateien, damit Sie die entwickelten Programme ggf. als Vorlage für die nächsten Laborübungen wiederverwenden können!