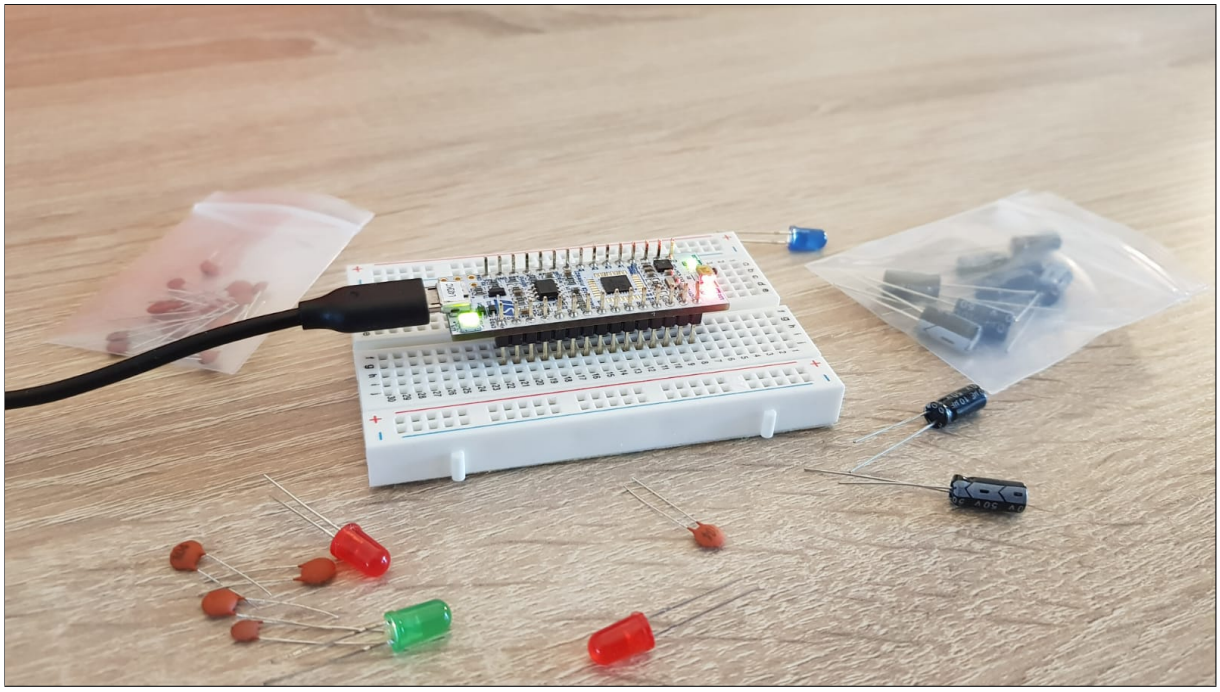
Hochschule für Technik und Wirtschaft Berlin
University of Applied Sciences

Computer Engineering

CEintro Beleg



NUCLEO-L432KC

Name: Hai Nam La

Matrikel: s0578105

7. Februar 2021

Inhaltsverzeichnis

1 Einführung

Im 11. Monats des Jahres 2020 haben wir Kommilitonen wie erwartet ein Termin zur Inbetriebnahme eines Nucleo-Boards bekommen. Aufgrund der aktuellen Lage wurden wir in Gruppen aufgeteilt um unsere Boards abzuholen, wobei mein Termin der 30.November 2020 um 12 Uhr war. Mit dem Nucleo-Board, NUCLEO-432KC, sollten wir dann mithilfe einer Aufgabenstellung [**1**] 3 verschiedene Prozesse programmieren und ausführen lassen. Dabei konnten wir uns unsere Entwicklungsumgebung selbst auswählen, wobei die über Linux mithilfe des STM32CubeMX empfohlen wurde. Außerdem muss der finale Quellcode auf einem eigen erstellten Git-Repository hochgeladen werden.

Die erste Aufgabe war es das LD3 zum blinken zu bekommen, die zweite Aufgabe war es, dass das Nucleo-Board über ein Terminal periodisch "Hello World“ wiedergibt und die dritte und letzte Aufgabe war es eine Eingabe über ein Terminal vom Nucleo-Board im Morse übersetzen zu lassen und diese mit dem LD3 wiederzugeben im Form von kurz anhaltenden blinken “.“ und lang anhaltenden blinken “-“.

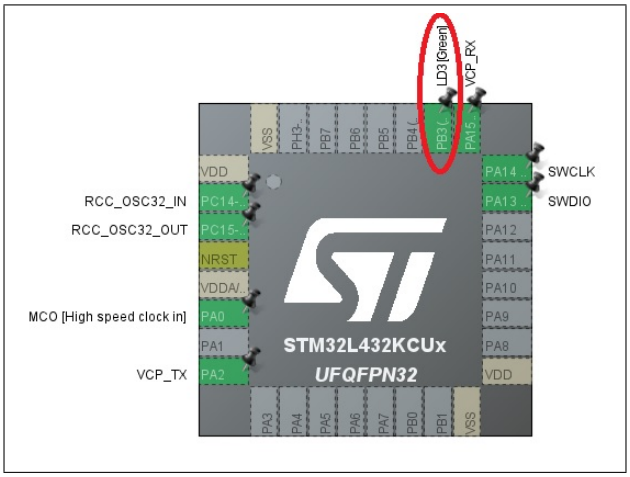
2 Herangehensweise

In der Begleitübung der selben Woche, als wir das Board bekommen haben, haben wir mit Anweisung von Prof. Dr. Sebastian Bauer angefangen uns die Entwicklungsumgebung zu erschaffen. Wir arbeiteten dabei mit Linux und folgten seinen Anweisungen, wobei die auch in der Aufgabenstellung[**1**] steht. Natürlich hätten wir auch vorher anfangen können oder mit dem STMCubeIDE über Windows arbeiten können. Darauf folgend setzten wir uns individuell an die Aufgabe, wobei einzelne Begleitübungen folgten indem wir Unterstützung von Prof. Dr. Bauer und sogar anderen Kommilitonen bekommen haben.

Für die erste Aufgabe musste ich mich erst vertraut machen mit dem Quellcode beziehungsweise mit dem Quellcodegerüst des Nucleo-boards. Viele der Sachen die zum Beispiel im Quellgerüst stehen, waren für uns von keiner Bedeutung. Da ich kaum bis keine Information im Internet zum Board fand bis auf gelöste Aufgaben, entschied ich mich der Aufgabenstellung[**1**] entlang zu arbeiten.

Die erste Aufgabe war das Blinken des LD3 vom Nucleo-Board, genannt “Blinky“. Als erstes musste mithilfe vom STM32CubeMX ein Quellgerüst erstellt werden. Dieses habe ich dann mit meinem Git-Repository gelinkt um Fortschritte hochladen zu können. Um das LD3 jetzt blinken zu lassen musste man in einer “While 1“ Schleife arbeiten um das Blinken zu periodisieren. Da das Quellcodegerüst schon eine While-Schleife gegeben hat, habe ich darin gearbeitet. Die Aufgabenstellung[**1**] gab die “HAL_GPIO_TogglePin”’ und die “HAL_Delay“ zur Verfügung, wobei die Erste einen Pin ansteuert und die Zweite eine Art Frequenz bestimmt.

Um die Argumente für die “HAL_GPIO_TogglePin”’ herauszufinden habe ich mir die grafische .ioc Datei des STM32CubeMx angesehen und herausgefunden, dass die LD3 durch den Pin “B3“ angesteuert wird



und die als Argument im Code benutzt, mit einem Delay von einer Sekunde.

```
 96   /* USER CODE BEGIN WHILE */
 97   while (1)
 98   {
 99       HAL_GPIO_TogglePin(GPIOB, GPIO_PIN_3);
100       HAL_Delay(1000);
101     /* USER CODE END WHILE */
```

Die zweite Aufgabe war es periodisch “Hello World“ in einem Terminal ausgeben zu lassen. Dabei habe ich einem String “helloworld“ mit der Zeichenkette “Hello World.“ belegt und musste nur noch mit dem von der Aufgabenstellung[**1**] gegebene “HAL_UART_Transmit", dem Nucleo-Board die Anweisung geben, den String auszugeben, wobei das “huart2“, beziehungsweise das USART2, welches als Kommunikation zwischen den Nucleo-Board und einem seriellen Port dient, angesprochen wird. Dabei wird noch ein Delay von einer Sekunde hinzugefügt um dadurch die periodische Ausgabe zu simulieren.

```
 96   /* USER CODE BEGIN WHILE */
 97   while (1)
 98   {
 99       char helloworld[]="Hello World.\n\r";
100       HAL_UART_Transmit(&huart2,helloworld,strlen(helloworld),1000);
101       HAL_Delay(1000);
102
103     /* USER CODE END WHILE */
```

Die dritte Aufgabe besteht darin, dass wir Zeichenketten im Morsecode auslesen lassen sollen. Zuallererst erstellte ich einen String “morse“ und definierte die einzelnen Buchstaben, Groß- und Kleinschreibung, die Zahlen 0-9 und einige Sonderzeichen mit ihrem Morse-Gegenstück.

```
static const char morse [][8] = {
        ['A'] = ".-",
        ['B'] = "-...",
        ['C'] = "-.-.",
        ['D'] = "-..",
        ['E'] = ".",
        ['F'] = "..-.",
        ['G'] = "--.",
        ['H'] = "....",
        ['I'] = "..",
        ['J'] = ".---",
        ['K'] = "-.-",
        ['L'] = ".-..",
        ['M'] = "--",
        ['N'] = "-.",
        ['O'] = "---",
        ['P'] = ".--.",
        ['Q'] = "--.-",
        ['R'] = ".-.",
        ['S'] = "...",
        ['T'] = "-",
        ['U'] = "..-",
        ['V'] = "...-",
        ['W'] = ".--",
        ['X'] = "-..-",
        ['Y'] = "-.--",
        ['Z'] = "--..",
        [' '] = " ",

        ['1'] = ".----",
        ['2'] = "..---",
        ['3'] = "...--",
        ['4'] = "....-",
        ['5'] = ".....",
        ['6'] = "-....",
        ['7'] = "--...",
        ['8'] = "---..",
        ['9'] = "----.",
        ['0'] = "-----",

        ['.'] = ".-.-.-",
        [','] = "--..--",
        ['?'] = "..--..",
        ['/'] = "-..-.",
        [':'] = "---...",
        ['+'] = ".-.-.",
        [';'] = "-.-.-.",
        ['-'] = "-....-",
        ['='] = "-...-",
};
```

Danach richtete ich mich nach der Aufgabenstellung und nutzte sowohl die “HAL_UART_Transmit“ als auch die “HAL_UART_Receive“, doch ich kam auf kein funktionierendes Ergebnis. Ich setzte mich mit einem Kommilitonen zusammen und wir haben zusammen eine Lösung gesucht. Uns kam nach einiger Zeit und Überlegung ein Code mithilfe des ASCII-Values, wobei wir nur die Kleinbuchstaben im String “morse“ definieren mussten

```
        for (int j = 0; j < count; j++)
        {

          if (Rx_data[j] == '\r' || Rx_data[j] == '\n' )
          {
          }

          if (Rx_data[j] >= 97 && Rx_data[j] <= 122)
          {
              ascii_value = Rx_data[j] - 32;
          }
          else
          {
              ascii_value = Rx_data[j];
          }

          zahlen[j] = (char)ascii_value;
        }
```

und haben die einzelnen Stellen des Arrays “Rx_data“, welches die Eingabe des Terminals übernimmt, auslesen lassen und diese mit dem langen Blinken und dem kurzen Blinken der LD3, den Morsecode imitiert.

```
185         for (int k = 0; k < count; k++)
186         {
187             for (int i = 0; i < 8; i++)
188             {
189                 current_letter = (char)morse[zahlen[k]][i];
190                 if (current_letter == '.' )
191                 {
192                   HAL_GPIO_WritePin(LD3_GPIO_Port, LD3_Pin, GPIO_PIN_SET);
193                   HAL_Delay(100);
194                   HAL_GPIO_WritePin(LD3_GPIO_Port, LD3_Pin, GPIO_PIN_RESET);
195                   HAL_Delay(100);
196                 }
197                 if (current_letter == '-' )
198                 {
199                   HAL_GPIO_WritePin(LD3_GPIO_Port, LD3_Pin, GPIO_PIN_SET);
200                   HAL_Delay(300);
201                   HAL_GPIO_WritePin(LD3_GPIO_Port, LD3_Pin, GPIO_PIN_RESET);
202                   HAL_Delay(100);
203                 }
204                 if (current_letter == ' ' )
205
206                 {
207                   HAL_GPIO_WritePin(LD3_GPIO_Port, LD3_Pin, GPIO_PIN_RESET);
208                   HAL_Delay(700);
209                 }
210
211                 }
212             }
213         }
```

Jetzt gibt es noch das Problem, dass zum einen die Eingabe sehr verzögert ist, was man beheben kann indem man die Timeout-Value des “HAL_UART_Transmit“ und des “HAL_UART_Receive“ anpasst und das Problem, dass die letzte Stelle des Arrays durch die While-Schleife ewig in einer Schleife steckt. Leider fanden wir keine Lösung, aber durch Recherche wurde es möglich mithilfe von “printf“ und “scanf“ zu arbeiten, welches das Problem löste. Dabei nutzte ich eine erstellte library “retarget.h“, die von “retarget.c“ beschrieben wird, die “printf“ und “scanf“ definiert[**2**]

```
4  #ifndef __RETARGET_H__
5  #define __RETARGET_H__
6
7  #include "stm32l4xx_hal.h"
8  #include <sys/stat.h>
9
10 void RetargetInit(UART_HandleTypeDef *huart);
11 int _isatty(int fd);
12 int _write(int fd, char* ptr, int len);
13 int _close(int fd);
14 int _lseek(int fd, int ptr, int dir);
15 int _read(int fd, char* ptr, int len);
16 int _fstat(int fd, struct stat* st);
17
18 #endif //#ifndef  RETARGET H
```

und konnte so mit wenigen Anpassungen des Codes und meiner Programmiererfahrung die Aufgabe lösen, wobei ich die library “retarget.h“ und “retarget.c“[**2**] in den Codegerüsten

“main.h“ und “stm32l4xx_hal_msp.c“ geschrieben habe, da ich keine Informationen gefunden hab wie man eine Library erstellt für das STM-Quellgerüst.

```
  while (1)
  {
          printf(" ");
          scanf("%s", Rx_data);

          for (int j = 0; j < count; j++)
      {

        if (Rx_data[j] == '\r' || Rx_data[j] == '\n' )
        {
        }

        if (Rx_data[j] >= 97 && Rx_data[j] <= 122)
        {
            ascii_value = Rx_data[j] - 32;
        }
        else
        {
            ascii_value = Rx_data[j];
        }

        zahlen[j] = (char)ascii_value;
      }
      for (int k = 0; k < count; k++)
      {
                char c = Rx_data[k];
          printf("%c", c);
          for (int i = 0; i < 8; i++)
          {


                current_letter = (char)morse[zahlen[k]][i];
              if (current_letter == '.' )
              {
                HAL_GPIO_WritePin(LD3_GPIO_Port, LD3_Pin, GPIO_PIN_SET);
                HAL_Delay(100);
                HAL_GPIO_WritePin(LD3_GPIO_Port, LD3_Pin, GPIO_PIN_RESET);
                HAL_Delay(100);
              }
              if (current_letter == '-' )
              {
                HAL_GPIO_WritePin(LD3_GPIO_Port, LD3_Pin, GPIO_PIN_SET);
                HAL_Delay(400);
                HAL_GPIO_WritePin(LD3_GPIO_Port, LD3_Pin, GPIO_PIN_RESET);
                HAL_Delay(400);
              }
              if (current_letter == ' ' )

              {
                HAL_GPIO_WritePin(LD3_GPIO_Port, LD3_Pin, GPIO_PIN_RESET);
                HAL_Delay(1000);
              }
              if (current_letter == '.' || current_letter == '-'  || current_letter == ' '){
                  transmit_data[counter] = current_letter;
                  counter = counter + 1;
              }
          }
counter = 0;
      }
      memset(Rx_data, 0, sizeof(Rx_data));
```

3 aufgetretene Probleme

Vom Anfang bis zum Ende war die Aufgabe für mich sehr umfangreich und neu. Da ich das Programmieren erst neulich, mit Beginn des Studiums, angefangen habe, hatte ich noch große Schwierigkeiten. Auch mit vergangener Zeit und den seminaristischen Lehrvorträgen in Programmieren fiel die Aufgabe mir schwer, da kaum mit dem gelernten Wissen gearbeitet wurde, wobei vor allem die letzte Aufgabe eine große Herausforderung war.

Auch den Beleg in LaTeX zu schreiben stellte mich vor Hürden. Allgemein die Benutzung von LaTeX habe ich verstanden, dennoch gibt es vereinzelte Probleme die ich nicht verstehe. Manchmal wenn ich zitiere und es im Literaturverzeichnis verlinke kommt eine “undefined“-error Nachricht und würde ich die Bilder mit Abbildungsunterschriften versehen, werden die Bilder chaotisch strukturiert, so dass es nicht mehr mit dem Text korrespondiert. Leider fand ich online keine Lösung, dennoch versuche ich mein Wissen über LaTeX in den nächsten 5 Jahren zu verbessern.

Literatur

[1] **Prof. Dr. Sebastian Bauer**. *gitlab.rz.htw-berlin.de/bauers/ce59-einfuehrung-in-ce-wise-2020/-/blob/main/ce-first-steps.pdf*

[2] **Shawn hymel**. *shawnhymel.com/1873/how-to-use-printf-on-stm32/*

Cnoviello. *github.com/cnoviello/mastering-stm32/tree/master/nucleo-f030R8/system*

[3] **STM32-L432KC User manual**.