


HITWK

Hochschule für Technik,
Wirtschaft und Kultur Leipzig



Implementierung des Spiels PONG auf einem AVR-Microcontroller

BELEG IM MODUL
EMBEDDED SYSTEMS II

Konrad Hermsdorf
Constantin Wimmer

Prüfer: Prof. Dr.-Ing. PRETSCHNER

Leipzig, den 19. Januar 2022

Kurzfassung

In diesem Beleg soll die Programmierung eines AVR-Mikrocontrollers am Beispiel des Spiels „Pong“ erläutert werden. Dabei wird vor allem auf die eingesetzten Software-Werkzeuge und die Struktur des C-Programms eingegangen. Der Quellcode wird mittels Crosscompilation auf dem Host in Maschinencode für den Mikrocontrollers übersetzt. Dieser Vorgang wird mittels Makefile automatisiert.

Selbstständigkeitserklärung

Hiermit erkläre ich, dass ich den vorliegenden Beleg selbständig verfasst und nicht anderweitig zu Prüfungszwecken vorgelegt habe. Es wurden nur die ausdrücklich benannten Quellen und Hilfsmittel benutzt. Wörtlich oder sinngemäß übernommenes Gedankengut habe ich als solches kenntlich gemacht.

Ort, Datum: _____

Unterschrift: _____

Inhaltsverzeichnis

| | |
|--|------------|
| Inhaltsverzeichnis | I |
| Abbildungsverzeichnis | III |
| Tabellenverzeichnis | 1 |
| 1. Umsetzung des Projektes | 3 |
| 1.1. Komponenten und Aufbau | 3 |
| 1.2. Entwicklungsumgebung | 4 |
| 1.3. Programmstruktur | 5 |
| 1.3.1. Funktionsumfang | 5 |
| 1.3.2. Modularisierung des Codes | 7 |
| 1.4. Automatische Maschinencodeerstellung mittels Makefile | 8 |
| 2. Zusammenfassung | 9 |
| A. Doxygen-Dokumentation | 11 |

Abbildungsverzeichnis

| | |
|--|---|
| 1.1. Aufbau des Projektes | 3 |
| 1.2. Oberfläche von Visual Studio Code | 5 |
| 1.3. Vereinfachter Ablauf des Programmes | 6 |

Tabellenverzeichnis

| | |
|--|---|
| 1.1. Pinbelegung | 4 |
| 1.2. Inhalt der Quelltextdateien | 7 |

1. Umsetzung des Projektes

1.1. Komponenten und Aufbau

Das Projekt wird auf einem Arduino Micro umgesetzt. Dieser ist ein Entwicklerboard, basierend auf dem Mikrocontroller ATmega32U4. Er wird mit 16MHz getaktet und 5V betrieben. Der Programmspeicher umfasst 32KB. Weiterhin besitzt der Controller 2.5KB SRAM als Arbeitsspeicher sowie 1KB EEPROM für dauerhaftes Speichern von Werten. Die Steuerung des Spiels erfolgt über Joysticks, dessen Position über Potentiometer in eine Spannung abgebildet wird. Die Anzeige erfolgt über ein 4-Zeilen Display (Modell 2004) mit parallelem Interface. Ein FC-113 I2C-Brückenchip wandelt die seriell über I2C gesendeten Displaykommandos in das parallel Interface um. Das Programm wird mittels Programmer EvUSBasp auf den Controller geladen. Im Folgenden sind die wichtigsten verwendeten Komponenten aufgezählt:

- Arduino Micro
- I2C Displayinterface FC-113
- 4-Zeilen LC-Display Modell 2004
- 2 × Potentiometer-Joystick
- EvUSBasp USB-Programmer

Abbildung 1.1 zeigt den Aufbau des Projektes.

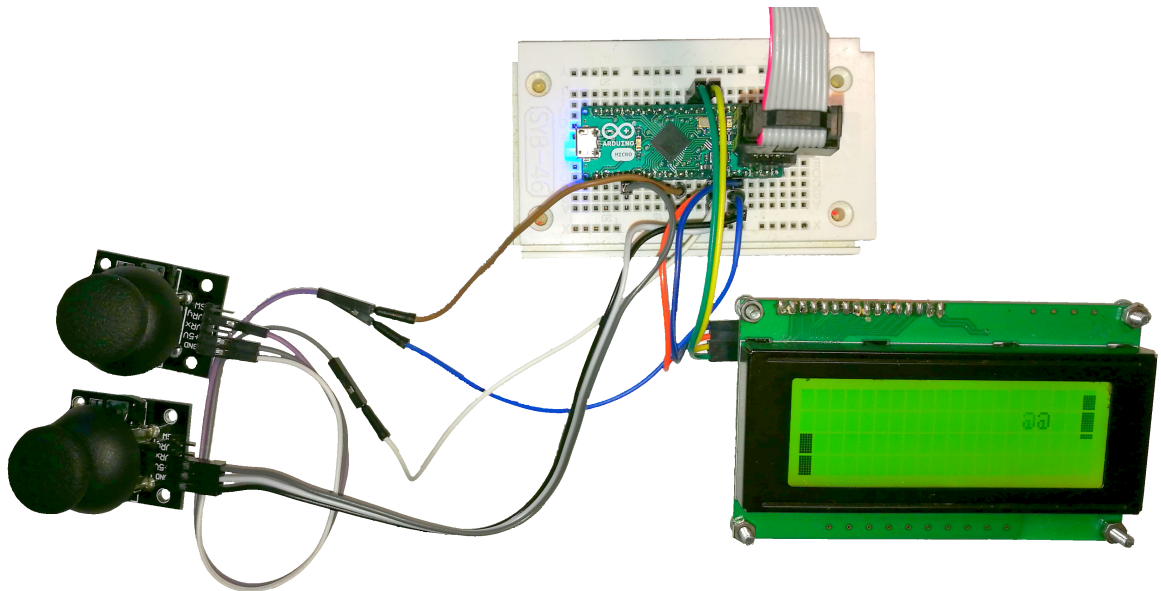


Abb. 1.1.: Aufbau des Projektes

1. Umsetzung des Projektes

Die nachfolgende Tabelle listet die Pinbelegung des Arduino Micro in diesem Projekt. Die Massepins und Versorgungsspannungen wurde für bessere Übersichtlichkeit ausgelassen.

Tabelle 1.1.: Pinbelegung

| Komponente | Pin am Arduino |
|-----------------------------|----------------|
| Joystick 1 - Teilerspannung | A5 |
| Joystick 2 - Teilerspannung | A0 |
| Display - I2C SDA | D2 |
| Display - I2C SCL | D3 |

1.2. Entwicklungsumgebung

Die zur Programmierung des ATmega32U4 unter Ubuntu Linux notwendigen Programme werden mittels Paketverwaltung „apt“ installiert:

- Es wird der ATmega-kompatible C-Compiler „avr-gcc“ genutzt.
- Für den Flash-Vorgang ist die Programmiersoftware „avrdude“ notwendig.
- Die automatische Kompilation und das Linken wird durch die Software „Make“ verwaltet. Dazu wird ein sog. „Makefile“ erstellt (siehe Abschnitt 1.4).
- Zur Erstellung des Quelltextes wird der Editor „Visual Studio Code“ eingesetzt.
- Der Programmer USBasp konnte ohne Installation von zusätzlichen Treibern eingesetzt werden.

Die Wahl des Editors fiel auf Visual Studio Code, da dieser Abhängigkeiten, sowie C-Syntax überprüfen kann. Die Integration von Intellisense als Algorithmus hinter der Autovervollständigung ermöglicht effizientes Verfassen von Quelltexten. Die nachfolgende Abbildung zeigt die Oberfläche von Visual Studio Code.

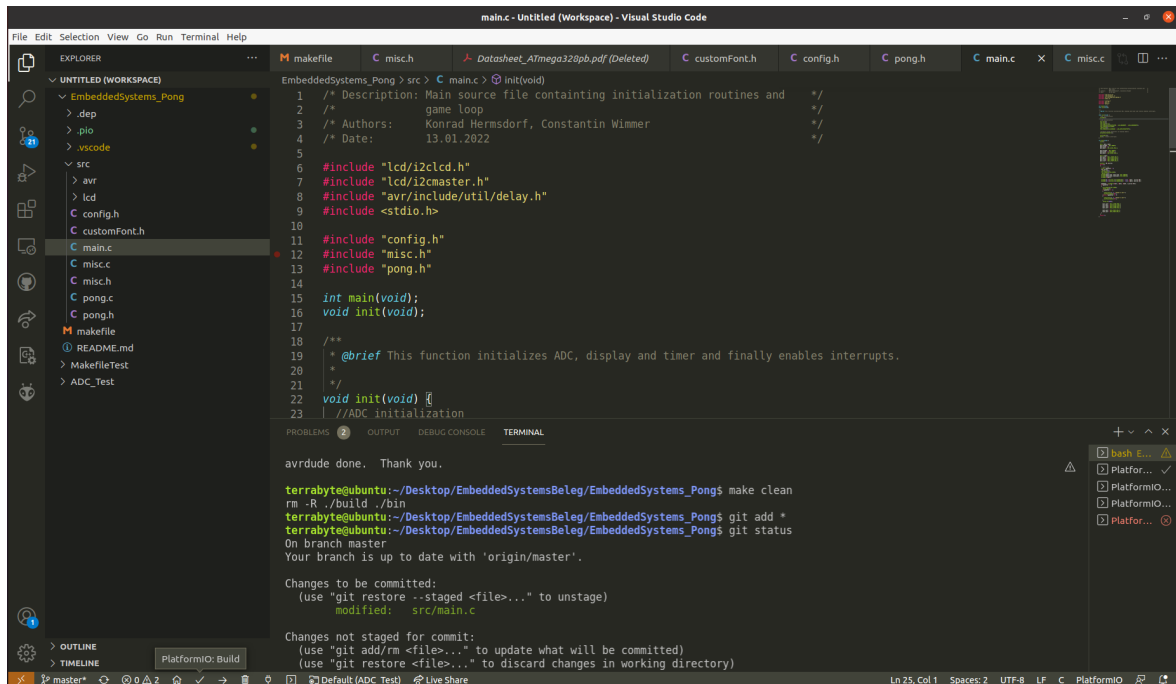


Abb. 1.2.: Oberfläche von Visual Studio Code

1.3. Programmstruktur

1.3.1. Funktionsumfang

Das Programm lässt sich grob in die Aufgabenbereiche Spielschleife, Zeichenroutinen und Verwaltung der Peripherie einteilen. Nach einer anfänglichen Initialisierungsphase, wird in der Spielschleife zyklisch die Position der Joysticks über den ADC-Wert abgefragt. Dieser Wert wird genutzt, um die Position der Ankerpunkte der Schläger im nächsten Bild zu berechnen. Bei Berechnung der Ballposition wird beachtet, ob der Ball hinter einem Schläger gelandet ist oder eine Wand berührt hat in erstem Fall wird eine Siegesnachricht angezeigt, in letzterem Fall die y-Richtung des Balls umgekehrt. Bei Berührung eines Schlägers wird die x-Richtung umgekehrt.

Die Größe der Bewegung je Bild für Ball und Schläger wird außerdem proportional zur verstrichenen Zeit zwischen zwei Bildern festgelegt. Dazu wird Timer 1 des ARmega32U4 zur Auslösung einer Interrupt-Service-Routine eingesetzt, welche eine Variable inkrementiert, die die Verstrichenen Mikrosekunden seit Timer-Start speichert.

Die Ansteuerung des Displays erfolgt über den Laborunterlagen entnommenen Programmcode. Dieser erlaubt die Programmierung eigener Schriftzeichen im Display. Das wurde genutzt um je Displayzeile 4 verschiedene Schlägerpositionen darstellen zu können. Eine ausführliche

1. Umsetzung des Projektes

Dokumentation aller Funktionen findet sich im Anhang A in Form einer Doxygen-generierten Quelltextbeschreibung. Der Ablauf des Programms ist vereinfacht in Abbildung 1.3 unter Vernachlässigung der Wartezeiten und Übergabewerte der Funktionen dargestellt.

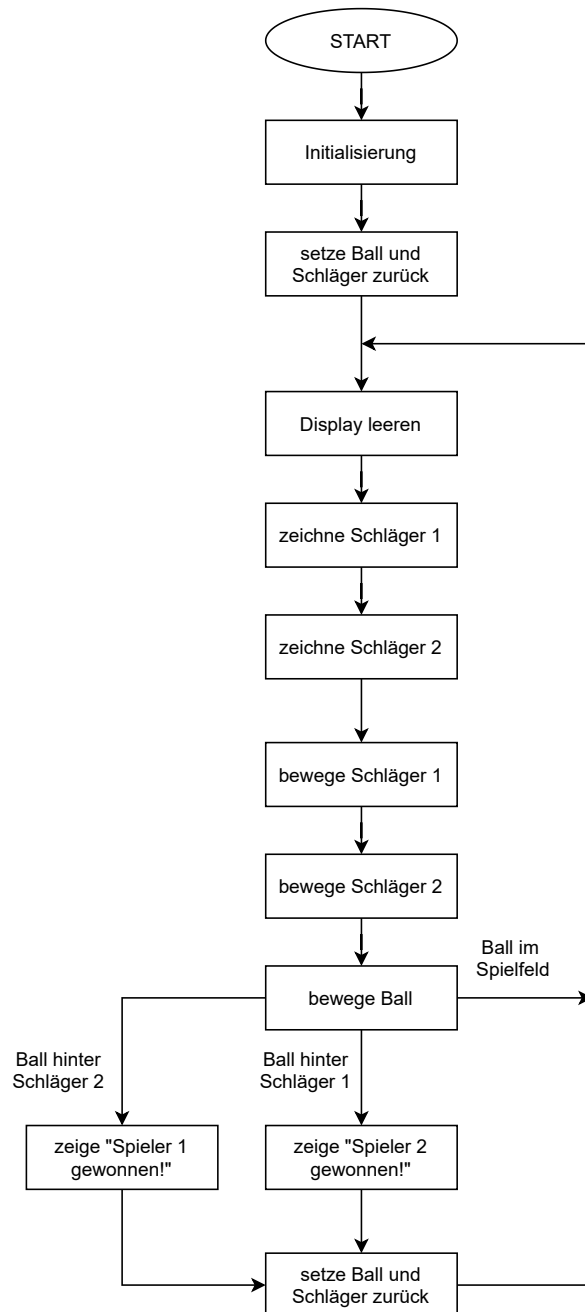


Abb. 1.3.: Vereinfachter Ablauf des Programmes

1.3.2. Modularisierung des Codes

Um die Wartbarkeit des Codes zu erhöhen, wird die Funktionalität in mehrere Teilfunktionen ausgelagert, welche wiederum thematisch in einzelne Quelltextdateien einsortiert werden. Tabelle 1.2 listet die Quelltextdateien und deren Inhalt. Sofern eine zur Headerdatei zugehörige .c-Datei existiert, bezieht sich die Tabelle auf beide Dateien.

Tabelle 1.2.: Inhalt der Quelltextdateien

| Quelltextdatei | Inhalt |
|---------------------------|---|
| <code>main.c</code> | <ul style="list-style-type: none"> • Initialisierung des Timers, Peripherie und Spielvariablen • Pong-Dauerschleife |
| <code>config.h</code> | <ul style="list-style-type: none"> • Präprozessormakros zur Konfiguration des Spiels • Adressen für Pins und Displayspeicher |
| <code>pong.h</code> | <ul style="list-style-type: none"> • Strukturvariablen für Ball und Schläger • Routinen zum Zeichnen des Spiels und Berechnen des Spielgeschehens |
| <code>customFont.h</code> | <ul style="list-style-type: none"> • Bitmuster der selbstdefinierten Zeichen für das Display |
| <code>misc.h</code> | <ul style="list-style-type: none"> • Funktionen zum Hochladen und Ausgeben der selbstdefinierten Zeichen auf das Display • ADC-Initialisierung und -Lesevorgang • Timer-Setup und zugehörige Interrupt-Service-Routine |

1. Umsetzung des Projektes

| | |
|--------------------------|--|
| <code>i2clcd.h</code> | <ul style="list-style-type: none">• bereitgestellter Quelltext• Grundlegende Routinen zum Ansteuern des Displays mit I2C-Expander |
| <code>twimaster.h</code> | <ul style="list-style-type: none">• bereitgestellter Quelltext• Routinen zur Kommunikation mit einem I2C-Slave, wie dem Display |

1.4. Automatische Maschinencodeerstellung mittels Makefile

2. Zusammenfassung und Ausblick

Es konnte gezeigt werden, dass der Mikrocontroller ATmega32U4 in der Lage ist, das Spiel „PONG“ flüssig auszuführen. Dabei wird auf Fähigkeiten des Controllers, wie Timer und Interrupts zurückgegriffen. Bei der Durchführung wurden Erkenntnisse über den Einsatz von Crosskompilation und der Erstellung von Makefiles zur Automatisierung des Buildvorgangs gewonnen. Als Erweiterung des Spiels wäre ein Zählen der Spieler-Punkte denkbar. Weiterhin könnte die Ballposition innerhalb der Displayelemente durch eigene einprogrammierte Zeichen, ähnlich der Schlägerposition, verfeinert werden.

A. Doxygen-Dokumentation