Übung Nr. 6

Jahrgang: BHME20
Gruppe: 3D

Protokollabgabe
Solldatum: 6.6.2024
Istdatum: 6.6.2024
Note:

PROTOKOLL

Thema: Programmierung und Steuerung von LEDs am Arduino MEGA über den seriellen Monitor

Tag: 16.05.2024
Zeit: 10:45-13:15 Uhr
Ort: HTBLA Kaindorf | PRR Labor
Anwesend: Traußnigg Jan, Christopher Wack, Unterberger Peter
Abwesend: Alex Uhl
Schriftführer*in: Traußnigg Jan
Betreuer: Dipl.-Ing. Steiner Walter

Aufgabenstellung

Ein Programm schreiben, das Eingaben vom seriellen Monitor verarbeitet und basierend auf diesen Eingaben LEDs am Arduino MEGA steuert.

Resümee

Während des Entwicklungsprozesses traten Probleme beim Kompilieren und Arbeiten mit dem seriellen Monitor der PlatformIO-Extension in VSCode auf. In zukünftigen Einheiten planen wir, eine andere Extension für den seriellen Monitor zu verwenden und vorgefertigte Bibliotheken zu integrieren.

Unterschriften

Christopher Wack

Jan Traußnigg

Peter Unterberger

Inhaltsverzeichnis

1 Zeitplan

10:45 – 11:35 Erklärung zu Digitalem Zwilling, Telegramm
11:35 – 13:15 Praktische Übung

2 Thema

2.1 Detaillierte Aufgabenstellung

Ein Programm soll entwickelt werden, das Zeichen vom seriellen Monitor empfängt und nach Drücken der "Enter"-Taste die gesammelten Zeichen als String speichert. Dieser String wird dann verwendet, um Befehle am Arduino MEGA Board auszuführen. Beispielsweise soll der Befehl "SET LED 1" die erste LED auf dem Board zum Leuchten bringen. In zukünftigen Einheiten sollen weitere Funktionen programmiert werden.

2.2 Verwendete Geräte und Hilfsmittel

- Entwicklungsumgebung: Visual Studio Code (VSCode) mit der PlatformIO-Extension
- Kompilierung und Upload: PlatformIO
- Versionsverwaltung und Datensicherung: Git und GitHub
- Hardware: Arduino ATmega2560
- Von Herrn Professor Steiner bereitgestelltes Board mit Tastern, LEDs, einem Display und weiteren Bauteilen

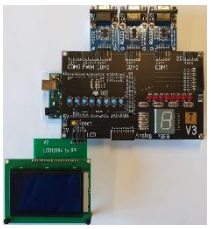

Abbildung 1: Arduino Board

2.3 Wichtige Erklärung

- **Serial.begin**: Initialisiert die Datenrate für die serielle Datenübertragung.
- **Serial.read**: Liest eingehende serielle Daten ein (Zeichen der Tastatur).
- **Serial.write**: Sendet einen String an den seriellen Monitor.
- **Serial.available**: Gibt an, wie viele Zeichen verfügbar sind.
- **Serial.event**: Wird aufgerufen, wenn ein Zeichen empfangen wird.
- **strcpy**: Kopiert den Inhalt eines Arrays in ein anderes Array.

3 Vorgangsweise

- Grundlagen der Datenübertragungsprotokolle und zugehörigen Telegramme besprochen.
- Programmstruktur für die serielle Kommunikation und LED-Steuerung erarbeitet.
- Programmierung und Testen des Codes in VSCode mit PlatformIO.
- Analyse und Lösung der Kompilierungsprobleme sowie der Herausforderungen mit dem seriellen Monitor.

3.1 Messergebnisse

```
#include <Arduino.h>
#define MAXSTR 80  // Maximale Länge der empfangenen Zeichenkette

// Definition der Pins für die LEDs
int ledPin[] = {49, 48, 47, 46, 45, 44, 43, 42};

char rx0Str[MAXSTR] = "";  // Puffer für empfangene Zeichenkette
char *token = NULL;  // Zeiger für die Tokenisierung der Zeichenkette
bool rx0Flag = false;  // Flag zum Anzeigen, dass eine vollständige Zeichenkette empfangen wurde

void setup()
{
  Serial.begin(9600);  // Serielle Kommunikation mit 9600 Baud starten
  for(int j = 0; j <= 7; j++)
  {
    pinMode(ledPin[j], OUTPUT);  // Konfiguration der LED-Pins als Ausgänge
  }
}

void loop()
{
  if (rx0Flag)  // Wenn eine neue Zeichenkette empfangen wurde
  {
    rx0Flag = false;  // Flag zurücksetzen
    Serial.println(rx0Str);  // Empfangene Zeichenkette zu Debugging-Zwecken ausgeben
    token = strtok(rx0Str, " ");  // Zeichenkette in Token zerlegen (erstes Token)

    if (token != NULL)  // Wenn das erste Token nicht NULL ist
    {
      if(!strcmp(token, "SET"))  // Überprüfen, ob das erste Token "SET" ist
      {
        if((token = strtok(NULL, " ")) != NULL)  // Nächstes Token (zweites Token) holen
        {
          if(!strcmp(token, "LED"))  // Überprüfen, ob das zweite Token "LED" ist
          {
            if((token = strtok(NULL, " ")) != NULL)  // Nächstes Token (drittes Token) holen
            {
              int i = atoi(token);  // Drittes Token in eine Ganzzahl umwandeln (LED-Index)
              Serial.println("TEST i");
              Serial.println("i");

              if((token = strtok(NULL, " ")) != NULL)  // Nächstes Token (viertes Token) holen
              {
                if(!strcmp(token, "1"))  // Überprüfen, ob das vierte Token "1" ist
                {
                  digitalWrite(ledPin[i], HIGH);  // Entsprechende LED einschalten

                  // Ausgabe zu Debugging-Zwecken
                  char buffer[50];
                  sprintf(buffer, "LED %d wird angesteuert, Pin %d", i, ledPin[i]);
                  Serial.println(buffer);
                }
                else if(!strcmp(token, "0"))  // Überprüfen, ob das vierte Token "0" ist
                {
                  digitalWrite(ledPin[i], LOW);  // Entsprechende LED ausschalten
                }
              }
              if(!strcmp(token, ""))  // Überprüfen, ob das vierte Token leer ist (ToDo)
              {
                // ToDo
              }
              else if(!strcmp(token, "OFF"))  // Überprüfen, ob das vierte Token "OFF" ist (ToDo)
```

```
              {
                // ToDo
              }
            }
          }
        }
      }
    }
  }
}

void serialEvent()
{
  // Vorgegebene Funktion aus dem Skript
}
```

- **Funktionalität**: Die grundlegende Funktion zur Zeichenaufnahme und Befehlserkennung funktionierte.
- **Probleme**: Schwierigkeiten beim Kompilieren und der Verwendung des seriellen Monitors in der PlatformIO-Extension.
- **Lösung**: Einsatz einer separaten Extension für den seriellen Monitor in zukünftigen Einheiten.
- **Planung**: Integration vorgefertigter Bibliotheken aus vorherigen Einheiten in das Projekt.

Diese systematische Vorgehensweise und die geplanten Verbesserungen sollen dazu beitragen, die Funktionalität des Projekts zu erweitern und die Effizienz bei der Entwicklung zu steigern.