Prozedurale Programmierung

Aufgaben 8

Mikrokontroller

Entwicklungsumgebung

In der Übung wird Visual Studio Code verwendet, alternativ kann als Entwicklungsumgebung Arduino IDE benutzt werden. Beachten Sie, dass Arduino IDE für den MXChip-Board konfiguriert werden muss, benötigt wird außerdem ein MXChip-Driver. Den Driver **en.stsw-link009.zip** und weitere Hinweise (Datei ArduinoIDEInstallation.txt) finden Sie in OPAL.

Konfigurieren von **Arduino IDE**:

- Tragen Sie unter Datei/Voreinstellungen eine zusätzliche Boardverwalter URL: https://raw.githubusercontent.com/VSChina/azureiotdevkit_tools/master/package_azureboard_index.json
- Suchen Sie unter Werkzeuge/Board... /BoardVerwalter z.B. nach AZ und veranlassen Sie die Installation von MXChip - Microsoft Azure IoT Developer Kit
- Wählen Sie als Board MXCHIP AZ3166 und als Port COM3
- Über Sketch/Hochladen können Arduino-Sketches aus Mikrokontroller hochgeladen werden.

Für **Visual Studio Code** werden benötigt:

- Zwei Extensions: „Azure IoT Device Workbench“ (s. vscode-iot-workbench-0.16.0.vsix im OPAL) und Arduino für VS Code,
- MXChip Driver (**en.stsw-link009.zip**, finden Sie u.a. in OPAL) und
- Arduino CLI (ist auch im Arduino IDE – Paket enthalten).
- Arduino Extensions müssen wie folgt konfiguriert werden:
  Einstellungen/Erweiterungen/Arduino/Additional Urls:
  https://raw.githubusercontent.com/VSChina/azureiotdevkit_tools/master/package_azureboard_index.json
  Einstellungen/Erweiterungen/Arduino/Path:
  z.B. C:\\Program Files (x86)\\Arduino

Eine ausführliche Installationsanleitung finden Sie in der Datei „Anleitung VS Code MXChip Windows.pdf“ in OPAL.

Aufgaben 0 – Das erste Beispiel

1. Schließen Sie den Mikrokontroller an USB an.
2. Starten Sie Visual Studio Code und installieren Sie die erforderlichen Extentions. Arduino-Extension wird im MarketPlace angeboten, „Azure IoT Device Workbench“ ist aus der Datei vscode-iot-workbench-0.16.0.vsix (s. OPAL) über „Extension/Install from VSIX…“ zu installieren.
3. Überprüfen Sie die Einstellungen von „Arduino“ und „IoT Device Workbench“ und verändern Sie diese ggf:

- arduino.path: C:\\Program Files (x86)\\Arduino,
- arduino.additional Urls: https://raw.githubusercontent.com/VSChina/azureiotdevkit_tools/master/package_azureboard_index.json

Die Veränderungen können zusammen über das Editieren von settings.json (s. Beispieldatei im Übungsordner in OPAL), aber auch einzeln über Menü (Schalter) „Einstellungen“ vorgenommen werden.

4. Erstellen Sie über „Anzeigen/Befehlspalette/Azure IoT Device Workbench: Create Project…“ ein neues Projekt. Benennen Sie das Projekt, wählen Sie dabei „Arduino Platform“ und „MXChipIoT DevKit“ aus.
5. Über „Anzeigen/Befehlspalette/Arduino: Board Manager“ muss „MXChip - Microsoft Azure IoT Developer Kit“ installiert werden.
6. Zusammen mit dem Projekt wurde ein Arduino-Sketch erstellt: C:\Users\...\Documents\IoTWorkbenchProjects\projects\IoTproject\Device**device.ino** Editieren Sie den neu erstellten Sketch, in dem Sie den folgenden Quellcode eintragen:

```
#include "Sensor.h"
RGB_LED rgbLed;
void setup() {
}
void loop() {
  rgbLed.setColor(255, 255, 255);
  delay(1000);
  rgbLed.setColor(0, 0, 0);
  delay(1000);
}
```

7. Wählen Sie über „Anzeigen/Befehlspalette/Arduino: Select Serial Port“ einen vorgeschlagenen Port, z.B. COM3
8. Laden Sie den Sketch über „Anzeigen/Befehlspalette/Arduino: Upload“ auf das MC-Board hoch.
9. Des Weiteren werden zur Lösung der Aufgaben die Ein- und Ausgaben über den seriellen Monitor notwendig. Das Öffnen des Monitors ist über „Anzeigen/Befehlspalette/Arduino: Open Serial Monitor“ möglich.

Aufgaben 1 - LEDs, delay und Serielle Schnittstelle

1. Blinken Sie ein SOS Signal (…−−−…) mit der LED.
   Nutzen Sie die Mehrfarben-LED (Klasse RGB_LED). Um zwischen lang und kurz zu unterscheiden verwenden Sie unterschiedliche Pausen zwischen dem Ein- und Ausschalten und zum Ausschalten der LED die Methode turnOff.

2. (Zusatz) Entwerfen Sie ein Treppenhauslicht, dass nach dem Drücken von Button A (USER_BUTTON_A [1]) für 5 Sekunden aktiv bleibt. Mit Button B (USER_BUTTON_B) kann das Licht aber sofort ausgeschalten werden.

3. (Zusatz) Das Lesen von der seriellen Schnittstelle ist mit Hilfe des Objektes Serial aus dem seriellen Monitor möglich (mit der Funktion read ist u.a. das Lesen genau eines Zeichens, s. [2])
   Steuern Sie damit die Aktivierung der LED. Wenn ein R gesendet wird, soll LED rot aufleuchten, ein G und B aktivieren jeweils grünes bzw. blaues Licht, mit einem C lässt sich LED abschalten.

Aufgabe 2 - Sensoren

Gegeben ist folgender Arduino-Sketch zur Ausgabe der aktuellen Temperatur im seriellen Monitor (s. Datei temperatur.ino):

```
#include "HTS221Sensor.h"
DevI2C i2c(D14, D15); // Objekt der Helper-Klasse DevI2C
HTS221Sensor sensor(i2c); // Sensor Temperatur und Feuchtigkeit
float temperatur=0;

void setup() {
  //Serial.begin(9600);
  //Serial.begin(115200);
  sensor.init(NULL);
}

void loop() {
  sensor.enable();
  sensor.getTemperature(&temperatur);
  sensor.disable();
  sensor.reset();

  Serial.print("\nTemperature: ");
  Serial.print(temperatur);
  delay(1000);
}
```

1. Verändern Sie den gegebenen Sketch so dass die Daten des Temperatursensors / Drucksensors zeilenweise in regelmäßigen Zeitabständen im seriellen Monitor ausgegeben werden, z.B.

```
21.60,962.11
21.70,962.27
```

Zur Abfrage des Luftdrucks wird die Klasse LPS22HBSensor mit der Methode getPressure benötigt. Beachten Sie, dass Drucksensor-Objekt ebenfalls mit dem Objekt der Helper-Klasse initialisiert wird.

2. (Zusatz) Zeigen Sie die aktuelle Temperatur auf dem Display an. Mit den Buttons sollte zwischen Grad Celsius und Fahrenheit umgeschaltet werden können.

Aufgaben 3 (Zusatz) - Display

1. Senden Sie mit der seriellen Schnittstelle vom Rechner an den Controller Zeichenketten, die dann auf dem OLED Display angezeigt werden. Für die Klasse OLEDDisplay ist das Einbinden von <OledDisplay.h> erforderlich (s. [3]) .

Verwenden Sie zum Lesen die Funktion Serial.readString(), die ein Objekt der Klasse String liefert. Überprüfen Sie zuvor ob die Eingabedaten bereits vorliegen (Serial.available() > 0).
Zur Ausgabe auf den OLED Display mit dem Objekt Screen (Screen.print(charbuf)) wird als „charbuf“ ein char[] benötigt. Zur Konvertierung zwischen String und char[] kann z. B. die Methode toCharArray der Klasse String verwendet werden.

2. Implementieren Sie ein mathematisches Quiz, das kleine Aufgaben umfasst, die zusammen mit der Lösung von der seriellen Schnittstelle gesendet werden, z.B. “B 1 + 3 = 5“. Auf dem Display werden nur die Aufgaben (z.B. “1 + 3 = 5”) angezeigt. Die wahre Aussage soll durch das Betätigen des Buttons A und die falsche Aussage des Buttons B erkannt werden. Zwei Counter zählen die richtigen/falschen Ergebnisse, die aktuellen Werte der Counter werden in der dritten Zeile des Displays angezeigt.
   Zusatzaufgabe: die Aufgaben sollen nicht gesendet, sondern erzeugt werden mit Hilfe eines Zufallsgenerators.

3. Realisieren Sie die Logik einer Ampel. Nach dem Drücken eines der Buttons erscheinen die Phasen der Ampel (Grün - Gelb - Rot - Rot/Gelb) als Zeichenketten („gruen“, „gelb“, und/oder „rot“) auf dem Display in den entsprechenden Zeilen.

Weiterführende Informationen:

[1] https://jeremylindsayni.wordpress.com/2017/11/11/using-the-mxchip-az3166-azure-devkit-with-arduino-coding-with-pin-names-instead-of-numbers/ (Reservierte Arduino-Variablennamen)

[2] https://www.arduino.cc/reference/en/ (Arduino Programmiersprache)

[3] https://microsoft.github.io/azure-iot-developer-kit/docs/apis/display/ (IoT DevKit Dokumentation)

[4] https://microsoft.github.io/azure-iot-developer-kit/docs/apis/hts221/ (IoT DevKit Dokumentation)