Projekt: „ESP-Universalsteuerung“

****1. Projektziel definieren****

"Erstelle eine Micro Python-Firmware für den ESP32-WROMM-32, als IDE nutze ich Thonny, die Software soll folgende Eigenschaften besitzt.

- Die Software dient dazu mithilfe eines ESP32-Board unterschiedlichste Maschinen über eine Ablaufschrittkette zu Steuern. Der Vorteil von diesem Programm ist die Flexibilität der Schrittkette die von jedem Bediener über einen WEB-Browser im gleichen Netzwerk erstellt , erweitert oder geändert werden kann ohne die Programmierung z.B. mit Micro Python , Arduino DIE oder andere umfangreiche/aufwendige Programmiersprachen nutzen zu müssen. Ach ohne Programmierkenntnisse kann so jeder seinen eigenen Maschinenablauf erstellen.

Folgende Aufgaben muss das Programm erledigen.

- Einwahl in das in der „config.py“ Datei vorgegebene WLAN-Netzwerk.

- Erstellen **einer** zweiteiligen WEB-Oberfläche mit der „microdot*„ Web-Framework für MicroPython*, einmal zur Übersicht der Ein und Ausgänge und der Handsteuerung darunter ein Bereich zum Erstellen und Bearbeiten der Schrittkette.

- Laden der vorhandenen JSON-Schrittkette oder, sollte die Datei noch nicht vorhanden sein, erstellen der „chain.json“ Datei. Ist eine vorhandene „chain.json“ Datei nicht mehr kompatibel , aufgrund von Änderungen oder Erweiterungen der Micro Python-Firmware, muss sie mit einer leeren „chain.json“ Datei überschrieben werden.

- Abfragen der Ein und Ausgänge des ESP und darstellen auf dem oberen Teil der WEB-Seite.

- Abfragen der Aktionen auf der WEB-Seite einmal zum händischen schalten der Relais , zum händischen bewegen der Servos, abfragen der Buttons zur Bearbeitung der Schrittkette.

- Das Schalten der Ausgänge und Abfragen der Eingänge sollte asynchron zur restlichen Abarbeitung des Programms ablaufen um Verzögerungen zu vermeiden. „Timer“ in der Schrittkette dürfen nicht das komplette Programm lahmlegen.

- Ein wichtiger Punkt ist der geringe Speicher des ESP32,bei alle Funktionen und Web-Darstellungen muss so effizient und Speichersparend wie möglich gearbeitet werde.

-Für den WLAN-Zugang sind Folgende Daten in „config.py“ zu verwenden.

ssid = ' ESP-Netz'

password = '2101-0815'

-Für die von dir zu erzeugende Micro Python-Firmware werde zwei Dateien erstellt:

„config.py“ für den WLAN-Zugang und die Zuordnung der ESP-Pins zu den ein und Ausgängen.

„main.py“ enthält den „Micro Python“ – Code und den HTML-Teil zur Erstellung WEB-Seite.

Die Schrittketten-Speicherdatei „„chain.json“ wird von der „Micro Python-Firmware“ erstellt und

verwaltet

- Wichtige Information: Der zu erstellende Programmcode für die „main.py“ oder die „config.py“ wird immer komplett von die ausgegeben, niemals nur in einzelnen Code Segmente. Der Speicherverbrauch der „Micro Python-Firmware“ ist stets zu beachten, durch Änderung unnötig gewordene Code-Schnipsel sind sofort zu löschen.

****2. Hardware-Konfiguration auflisten****

Steuerung: Es handelt sich um ein ESP32 WROMM 32 mit 38 Pins.

Folgende Ein und Ausgänge und Variablen sind in der „config.py“ zu definieren:

```
   # Servos
   servo1 = machine.PWM(machine.Pin(22), freq=50)
   servo2 = machine.PWM(machine.Pin(23), freq=50)


# Analoge Eingänge
   adc1 = machine.ADC(machine.Pin(34))
   adc2 = machine.ADC(machine.Pin(35))


# Digitale Eingänge
      "E1": machine.Pin(15, machine.Pin.IN, machine.Pin.PULL_DOWN),
      "E2": machine.Pin(17, machine.Pin.IN, machine.Pin.PULL_DOWN),
      "E3": machine.Pin(18, machine.Pin.IN, machine.Pin.PULL_DOWN),
      "E4": machine.Pin(19, machine.Pin.IN, machine.Pin.PULL_DOWN),
      "E5": machine.Pin(21, machine.Pin.IN, machine.Pin.PULL_DOWN)


 # Ausgänge (Relais)
      "R1": machine.Pin(32, machine.Pin.OUT, value=0),
      "R2": machine.Pin(33, machine.Pin.OUT, value=0),
      "R3": machine.Pin(25, machine.Pin.OUT, value=0),
      "R4": machine.Pin(26, machine.Pin.OUT, value=0),
      "R5": machine.Pin(27, machine.Pin.OUT, value=0),
      "R6": machine.Pin(14, machine.Pin.OUT, value=0),
      "R7": machine.Pin(12, machine.Pin.OUT, value=0),
      "R8": machine.Pin(13, machine.Pin.OUT, value=0),
```

Ausgänge (Zylinder)

"Z1_Auf": machine.Pin(4, machine.Pin.OUT, value=0),

"Z1_Ab": machine.Pin(5, machine.Pin.OUT, value=0)

"Z2_Auf": machine.Pin(2, machine.Pin.OUT, value=0),

"Z2_Ab": machine.Pin(16, machine.Pin.OUT, value=0)

****3. Funktionale Anforderungen gliedern****

-Ablauf:**

1. Booten:

- Hardware initialisieren ().
- WiFi verbinden (aus config.py).
- „chain.json“ Datei laden

2. Hauptschleife:

- Ein- und Ausgange müssen laufend ausgelesen und gesetzt werden. Asynchron

Folgende Importe werden gebraucht.

“import network

import uasyncio as asyncio

import machine

import time

import ujson

import uos as os“

****4. „Anzeige und Handbedienung“, obere Teil der Webseite ****

Als Hauptüberschrift unserer gesamten Webseite steht der Name „ESP-Universalsteuerung“

(Es wäre gut mit einem Browser im dem Netzwerk unserer Steuerung die Web-Seite unter der Adresse „http:// ESP-Universalsteuerung“ aufrufen zu können.)

Darunter in einer Umrandung mit etwa ¼“ der Seitenhöhe werden alle Ein- und Ausgänge im „Ist - Zustand“ angezeigt:

Das ist der Bereich „**Hardware Übersicht“**

1)- Die Digitalen Ausgänge „R1“ – „R8“ wird dargestellt in 8 Felden jeweils mit der Zahl des Relay als Inhalt und je nach Zustand in Rot oder grüner Farbe. Durch Anklicken mit der Maus oder durch die Schrittkette kann er geschalter werden

2)- Die Digitalen Ausgänge "Z1_Auf", "Z1_Ab", "Z2_Auf", "Z2_Ab":: wird dargestellt in 4 Felden jeweils mit der Bezeichnung „Z1+“ , „Z1-“ , „Z2+“ , „Z2-“ als Inhalt und je nach Zustand in Rot oder grüner Farbe. Durch Anklicken mit der Maus oder durch die schrittkette kann er geschalter werden.

3)- Die Digitalen Eingänge „E1“ – „E5“ werden dargestellt in 5 Felden jeweils mit der Zahl als Inhalt und je nach Zustand in Rot oder grüner Farbe.

4)- Die Analogen Eingänge „adc1“ und „adc2“ werden dargestellt in 2 x 5steligen Wertefeldern jeweils mit dem am ESP ermittelten Wert als Inhalt .

5)- Die Servo Ausgänge „servo1“ und „servo2“ werden dargestellt in 2 x 4steligen Wertefeldern , der Inhalt kann durch Pfeile im Wertefeld mit der Maus oder durch die Schrittkette im Bereich der für Servos möglichen Einstellung geändert werden.

Der manuelle/händische Eingriff in die Ein- und Ausgänge des ESP haben Vorrang gegenüber der Schrittkette.

Die Art und Weise der Darstellung der Elemente in diesem Bereich ist maßgeblich für den Speicherverbrauch, hier muss gut nachgedacht werden sehr sparsam vorgegangen werden.

****5. „Schrittkettenanzeige und Bearbeitung“, untere Teil der Webseite ****

Hier beginnt der wichtigste Teil des Projekts.

Allgemeine Beschreibung der Schrittkette.

Die Schrittkette ähnelt einer Tabelle mit 7x Feldern, jeder neu erstellte Schritt fügt eine neue Zeile hinzu. Die Anzahl der Schritte ist nur durch den Speicher begrenzt. Jeder neue Schritt verlängert die Webseite nach unten. Unterhalb des letzten Schrittes befinden sich die Buttons zur Erstellung und bearbeiten der Schrittkette. Beim Starten der Schrittkette wird sie von oben nach unten abgearbeitet. Nach dem letzten Schritt endet die Schrittkette und kann wieder neu gestartet werden. Eine Laufende Schrittkette kann über einen Button „Abbruch“ beendet werden. Es gibt zwei grundsätzliche Typen von Schritten. Ein Schritttyp ist Beschriftet als „Aktion“ der andere Schritttyp ist Beschriftet als „Bedingung“. Die „Aktion“ Schritte schalten die ESP-Ausgänge, nach dem Schaltvorgang springt der Schritt sofort und ohne Pause in den nächsten Schritt. Die „Bedingung “s Schritte arbeiten wie ein „If then“-Befehle als Bedingung werden Digitale oder Analoge ESP-Eingänge sowie Timer oder Schleifen genutzt. Ist die Bedingung erfüllt springt die Kette in den nächsten Schritt. Jeder einzelne Schritt wird im JSON-Format mit 6 Variablen gemäß der Felder bearbeitet und gespeichert. Hier der genaue Aufbau und die Beschreibung aller möglichen Schritte.

Hier alle Möglichkeiten der Schritte in Tabellenform.

Schrittnummer	Schritttyp	Bezeichnung	Auswahl Auswahlbox	Aktion Auswahlbox	Wert	Löschen
10	Aktion	Digital	1 bis 8	Ein / Aus	-	Button
20	Aktion	Servo	1 und 2	-/-	Position	Button
30	Aktion	Zylinder	1 und 2	Rein / Raus	-	Button
40	Bedingung	Digital	1 bis 5	An / Aus	-	Button
50	Bedingung	Analog	1 und 2	größer / kleiner	Wert	Button
60	Bedingung	Warten	-/-	Min. / Sec.	Zeit	Button
70	Bedingung	Sprung	1 bis 500	-/ -	wie oft	Button

Es gibt insgesamt 7 verschieden Schritte, 3 x „Aktion“ Schritte und 4 x „Bedingung“ der Aufbau der einzelnen Schritte ist immer so wie in der Tabelle dargestellt. Um immer die gleiche Anzahl der Felder in dem JSON Format zu haben werden ungenutzte Felder z.B. mit einem „-„ gefüllt. Die Schrittnummer ist hier in der Tabelle nur zur besseren Beschreibung durchnummeriert und hängen bei den Schritten natürlich von der späteren Position in der Schrittkette ab.

1) Die Schrittkettennummer wird immer bei jedem neuen Schritt automatisch um 10 gegenüber den vorherigen Schritt erhöht. Der Hintergrund der Schrittnummern dient als Anzeige welcher Schritt aktiv ist, oben im Beispiel wäre Schritt z.B. Schrittnummer 60 Aktiv.

2) Das Feld „Schritttyp“ zeigt nur an welcher Typ es ist.

3) Die „Bezeichnung“ beschreibt welcher Bereich an Ein- oder Ausgänge geschaltet werden müssen.

4) „Auswahl“ ist immer ein Auswahlfeld der möglichen Ein oder Ausgänge usw. durch die Angaben der Pinbelegung in der Datei „config.py“ sowie den Feld „Bezeichnung“ und dieser „Auswahl“ seht genau welcher Pin am ESP gemeint ist. Es ist deine Aufgabe die Pins aufgrund der Angabe in der Schrittkette logisch zu verknüpfen.

5) „Aktion“ ist immer eine Auswahlbox gibt an was mit dem PIN passieren soll z.B. An/Aus die Aktion wird ausgeführt sobald der Schritt erreicht und aktiv ist.

6) „Wert“ ist für die Eingabe von Integer Zahlen gedacht möglich von 0 – 5000 um z.B. Zeiten oder Analoge werte anzugeben.

7) Ist kein Feld sondern ein Button jeder Schritt hat ihn und dient zum Löschen von Schritten aus einer Schrittkette.

Beispiel: Aktionen.

Beispiele 1: Digital Ausgang 3 (entspricht R3 = Pin(25)) wird eingeschaltet

10	Aktion	Digital	3	Ein	/	Button

Beispiele 2: Servo 2 (entspricht servo2 = Pin(23)PWM) wird auf Position 150 gestellt.

20	Aktion	Servo	2	/	150	Button

Beispiele 3: Digital "Z1_Auf" = Pin(25)) wird eingeschaltet

30	Aktion	Zylinder	1	Raus	/	Button

Beispiel: Bedingung

Beispiele 4: Wenn der Digital Eingang 4 ("E4" = Pin(19)) auf Strom liegt läuft der Schritt weiter.

40	Bedingung	Digital	4	An	/	Button

Beispiele 5: Wenn der Analoge 2 ("E4" = Pin(19)) kleiner oder gleich dem Wert 1250 ist läuft der Schritt weiter.

50	Bedingung	Analog	2	kleiner	1250	Button

Beispiele 6: Bei Erreichen des Schritt 60 (das Schrittzahlfeld wird grün) startet ein Timer, nach Ablauf der eingeben 60 Sekunden läuft die schrittkette weiter in den nächsten Schritt.

60	Bedingung	Warten	/	Sec.	60	Button

Beispiel 7: Bei erreichen von Schritt 70 springt das Programm zurück in den Schritt 20 bei jeden Sprung wird der Sprungzähler hier die 5 um ein -1 verringert ist der Sprungzähler kleiner als 1 geht es in den nachfolgenden Schritt. Dabei wird der Sprungzähler wieder auf den ursprünglichen wert, hier 5 zurückgesetzt

70	Bedingung	Sprung	20	/	5	Button

****6. Elemente für den Umgang mit der Schrittkette**

Unterhalb der Schritte befinden sich Button mit verschieden Funktionen die die Schrittkette betrifft.

Folgende Buttons werden zur Erstellung der jeweiligen Schritte Benötigt:

Die arte der Schritte ist entsprechen dem Namen der Knöpfe. Sollte selbsterklärend sein.

Button 1 / „Aktion Digital“

Button 2 / „Aktion Analog“

Button 3 / „Aktion Zylinder“

Button 4 / „Bedingung Digital“

Button 5 / „Bedingung Analog“

Button 6 / „Bedingung Warten“

Button 7 / „Bedingung Sprung“

Button 8 / „Speicher“ speichert die Schrittkette in der „chain.json“ Datei ab.

Button 9 / „Laden“ lädt die Schrittkette aus der „chain.json“ Datei zurück in die Web Seite

Beim dem starten des ESPs wird, sollte eine funktionierende „chain.json“ Datei vorhanden sein, wird sie automatisch in Web Seite geladen.

Button 10 / „Starten“ startet die geladene Schrittkette.

Button 11 / „Abbruch“ bricht die laufende Schrittkette ab , löscht den Schrittanzeiger und setzt alle Timer oder Sprung-Anweisungen auf Anfang zurück.

Button 12 / „Sortieren“ Um eine Möglichkeit zu haben zusätzliche Zwischenschritte einzufügen kann man die automatsch erstellte Schrittnummer überschreiben. Nach dem drücken von „Sortieren“ wird die Schrittkette in der richtigen Reihenfolge neu aufgebaut. Für die neue Reihenfolge ist die Höhe Schrittnummer maßgebend.

Ich hoffe meine Ausführungen waren verständlich und vollständig, wenn nicht bitte ich vor der Programmerstellung die fehlende Information zu erfragen. Wenn alle Unklarheiten beseitigt sind bitte immer den Kompletten Code erstellen. Bitte an den geringen Speicher des ESP-denken.

Hier noch zwei Fragen die schon oft aufgekommen sind.

Zu Frage: "Wie genau soll die Schrittnummerierung dynamisch gehandhabt werden, vor allem wenn Zwischenschritte eingefügt werden" Es gibt den Button 12, ist auf der letzten Seite Beschrieben. Das heißt die Schrittfolge wird durch drücken von "Button 12" in der Reihenfolge der Schrittnummern neu dargestellt. Kleine Nummer zu erst . So kann ich einen neuen Schritt einfügen indem ich eine Schrittnummer vergebe die zwischen zwei Schritten liegt. z.B. 1. Schritt hat die Nr. (10), 2. Schritt hat die Nr. (20), usw. Ich erzeuge einen neuen Schritt der bekommt automatisch die Nummer (30). Ich ändere dann die (30) auf (15) um. Es stehen dann untereinander . Schritt (10), Schritt(20), Schritt (15). Durch drücken von "Button 12" wird die Schrittkette neu, in absteigender Reinfolge, neu sortiert dargestellt. Also sieht es dann so aus. 1. Schritt hat die Nr. (10), 2. Schritt hat die Nr. (15), 3. Schritt hat die Nr. (20). Hast du das verstanden ? Zu frage 2) "Wäre es sinnvoll, genauere Angaben zum maximal möglichen Speicherverbrauch oder zur maximalen Anzahl an Schritten zu haben? Ja, sollten wir auf jeden Fall einbauen. zu Frage 3) "Eventuell könnte noch eine kleine Anleitung ergänzt werden, wie im Fehlerfall (z.B. bei einem Verbindungsabbruch) vorzugehen ist." darum sollten wir uns Kümmern wenn wir es getestet haben und wir wissen welche Fehler auftreten können. So, vielen Dank das dir meine Ausführung gefällt. Habe viel Zeit darein gesteckt.

Wollen wir loslegen und den Code zusammenbauen ?