Fachinformatiker für Anwendungsentwicklung

Dokumentation

Smart Home

Abgabetermin: 17.12.2023

Kolesnikow Nikita

Spahija Ardilon

Ausbildungsbetrieb

Berufliches Schulzentrum für Wirtschaft und Datenverarbeitung Stettiner Straße 1

97072 Würzburg

Inhaltverzeichnis

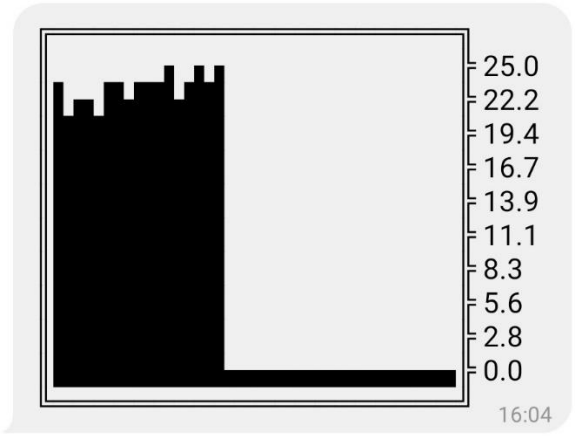
Inhalt

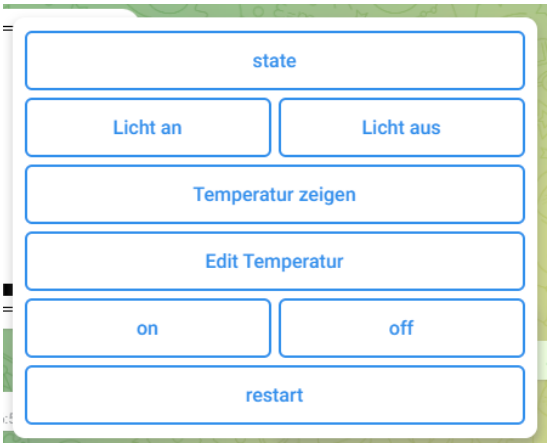
Was macht das Projekt?

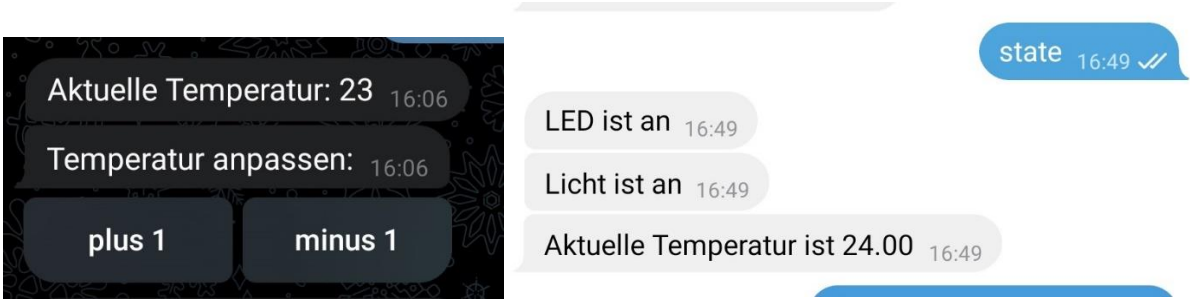
Das Projekt ist ein Smart Home System Konzept, welches innerhalb eines Lego-Gehäuses die einzelnen Bauteile beinhaltet. Ein RFID-Reader liest einen spezifischen RFID-Key ein, und bei Verwendung eines falschen Schlüssels wird eine sofortige Benachrichtigung an den Nutzer gesendet, über einen Telegram Chatbot, welches vom ESP8266 erstellt wurde.

Zusätzlich wird eine digitale Anzeige verwendet, um die Uhrzeit und Zimmertemperatur darzustellen. Durch die Integration eines Menüs im Telegram Chatbot kann der Nutzer folgendes: Status anzeigen, Licht in Zimmer an- und ausmachen, Temperatur in Grafik anschauen, Temperatur in Zimmer verwalten und die ESP8266 restarten.

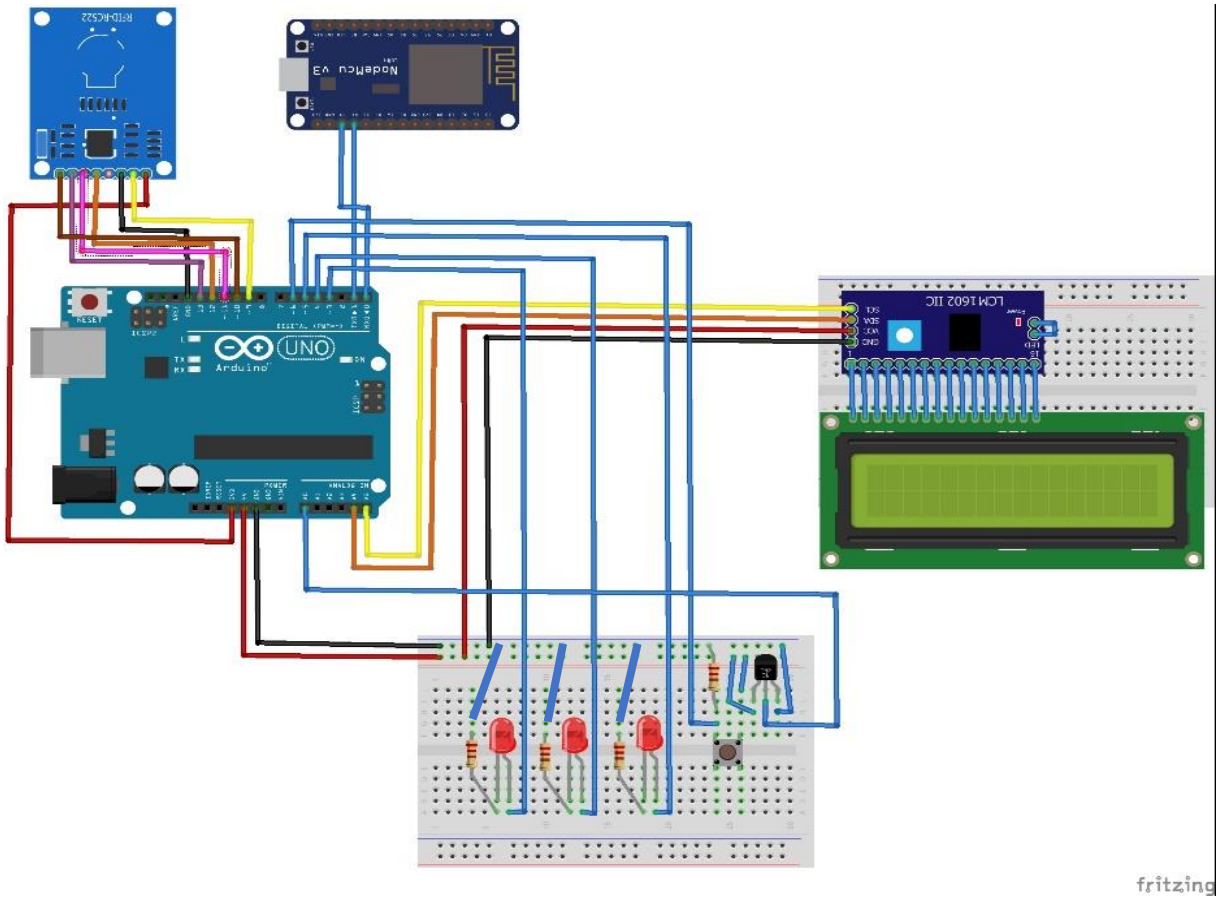
Die Temperaturmessung erfolgt präzise mittels eines TMP36, der dem System die Möglichkeit gibt, Echtzeitdaten zur Raumtemperatur zu erfassen. Es werden verschiedene Lichter benutz, das gelbe Licht Simuliert Licht in Zimmer, das Hellblaue Licht ist für die Simulation von Kühlanlage und das Rote licht ist für Heizanlage.







Aufbau des Projektes



Der RFID-RC522 verwendet die Pins 3.3V, GND, 13, 12, ~11, ~10 und ~9 vom Arduino UNO. Die Verbindungen zwischen den Pins und dem RFID-Modul erfolgen spezifisch: 3.3V zu 3.3V, RST zu ~9, GND zu GND, MISO zu 12, MOSI zu ~11, SCK zu 13 und SDA mit ~10.

Ein Breadboard, das für LEDs und einen TMP36 verwendet wird, benötigt den UNO, um die V5 Rail und den GND Rail zu verbinden. Hierbei wird der V5 Pin mit dem V5 Rail und der GND Pin mit dem GND Rail verbunden. Der Knopf erfordert einen 220 Ohm Widerstand, der am GND Rail angeschlossen ist. Die drei LEDs benötigt je einen 220 Ohm Widerstand und ein Kabel, wobei die Kabel mit den Pins ~5, 4 und ~3 des UNO verbunden werden. Der TMP36 benötigt drei Kabel für +5V, Analog Out und GND. Zwei der Kabel werden am V5 Rail und GND Rail angeschlossen, das dritte am UNO A0 Pin.

Ein weiteres Breadboard wird für den LCM 1602 IIC und den LCD 1602 verwendet. Der LCM wird mit den Pins A4 und A5 des UNO sowie dem V5 Rail und GND Rail des ersten Breadboards verbunden. Der GND des LCM wird mit dem GND Rail und der VCC mit dem V5 Rail verbunden. Beim UNO wird der SDA mit A4 und der SCL mit A5 verbunden.

Schließlich wird der esp8266 benötigt, um mit dem Telegram Chatbot zu kommunizieren. Der esp8266 wird mit dem UNO verbunden, und werden über UART kommuniziert.

Erläuterung und Beschreibung des Codes

Zuerst wird Code für Arduino Uno erläutert und beschrieben. Arduino Uno verwendet folgende Bibliotheken: LiquidCrystal_I2C.h, RFID.h, und GyverOS.h. LiquidCrystal_I2C.h wird benötigt, um mit LCD-Display über I2C-Schnittstelle mit Arduino zu verbinden. Dazu braucht man auch LCM 1602 IIC. RFID.h ist Bibliothek für RFID-reader. Und die GyverOS.h ist ein einfache Task Manager, diese Bibliothek wird später insbesondere angeschaut.

Als nächstes wird Pin A0 für Temperatur Sensor, variablen rawVoltage, voltage und currentTemp angelegt. Für RFID-reader werden SS-PIN und RST_PIN definiert, die Übergabe Parameters für RFID sind. LiquidCrystal_I2C wird als lcd angelegt. Danach werden zwei Schlüsseln angelegt, und ein String rfidCard für Schlüssel, der abgeglichen wird. Und jetzt wird Task Manager angelegt, es sieht folgendes aus: GyverOS<6> OS;. Zahl 6 bedeutet das Arduino wird 6 Aktionen (gleichzeitig) erledigen. Diese Bibliothek wurde benutzt aufgrund das die Arduino Uno kann gleichzeitig nur eine Aufgabe erledigen. Anschließend werden angelegt variable now für Berechnung von Zeit, h und m für Stunden und Minuten, und bool wrongKey.

Jetzt kommen wir zum void setup(). Als erstes wird Serial.begin mit Bankbreite 9600 initialisiert, um Kommunikation über UART zu ermöglichen. Da werden auch lcd und rfid initialisiert, Hintergrund Beleuchtung bei Display angemacht, und alles bei Display gelöscht. Pins 3, 4, 5 sind für Output. Von dieser Stelle wird Funktion temperatur() aufgerufen. Und hier werden Aufgaben für Task Manager angegeben, Arduino hat folgende Funktionen, die die erledigen muss: temperatur, rfidRader, clock, serialReader, getTime und sendData. Diese Aufgaben sind Funktionen die Arduino in Zyklus erledigen wird. Zum Beilspiel: OS.attach(0, temperatur, 10000);, erste Übergabeparameter ist Nummer von die Funktion die immer von 0 beginnen. Zweite Übergabeparameter ist Name von der Funktion, und als drittes schreibt man wie oft der Funktion aufgerufen soll. Aufruf ist in der Funktion loop und, sieht folgendes aus: OS.tick();.

Danach kommen die Funktionen, die angelegt wurden. Erste Funktion ist void temperatur(), und da wird Temperatur Sensor die Voltage messen und mithilfe von eine Formel in Temperatur umwandeln und in die variable currentTemp gespeichert. Gespeicherte Temperatur wird an Position 12, 0 bei Display ausgegeben. Anschließend wird die Temperatur in Zimmer mit Temperatur aus Datenpaket von ESP8266 verglichen.

Jetzt kommt die Funktion rfidReader, die wird jede 1,5 Sekunden aufgerufen. Es beginnt, wenn RFID-Reader ein Schlüssel findet. In Variable rfidCard wird Serial Nummer gespeichert, der aus vier teile besteht. Danach prüft die Arduino, ob der Schlüssel mit ein von beiden angelegten Schlüsseln übereinstimmt. Falls Schüssel passt, wird am Display geschrieben,

dass Schlüssel passt für drei Sekunden. Falls es nicht der Fall ist, wird am Display geschrieben, dass Schlüssel nicht passt. Um eine Benachrichtigung von Telegram Bot zu kriegen, wird variable wrongKey auf true gesetzt. Nach diese If Else Statement wird wieder Temperatur gemessen, weil die vorher gelöscht wurde.

Nächste kleine Funktion ist clock(), die ist zuständig für Ausgabe von Zeit. Ein wichtiger Hinweis ist, dass die Zeit wird, erst richtig angestellt, wenn die Arduino eine Nachricht mit Schlüssel „111“ kriegt. Zeit wird in Funktion getTime() gerechnet mithilfe eine Formel. Um die Stunden nicht weiter als 24 und Minuten nicht weiter als 60 wurde Modulo benutzt.

Weiter kommt Funktion serialReader(), das ist eine wichtigste Funktion. Über diese Funktion kriegt Arduino Uno Aktuelle Zeit, Temperatur, die in Zimmer beibehaltet werden muss, und Signal für Licht. Die funktioniert folgendes: als erstes wird Serial gelesen und falls dort was steht, wird das bis Zeichen „;“ gespeichert, und splitet diese Zeile und wandelt das gleich in Integer. Danach wird erste Wert mit Index 0 geprüft, wenn das erster Wert ist „111“ dann es ist das richtige Paket und es liest weiter. Nächste Zwei Werte sind Stunden und Minuten, die werden gleich in variablen h und m gespeichert. Vierter Wert ist An- und Ausschalter für das Licht. Und letzter Wert ist Temperatur. Insgesamt sieht dieses faket folgendes aus: 111,Stunden,Minuten,1/0,Temperatur;.

Um Nutzer eine Nachricht krieg das es wurde falsche Schlüssel, benutzt oder Temperatur in Grafik anschauen, muss die Esp8266 von irgendwo Daten kriegen. Das macht Die Funktion sendData(). Die hat Schlüssel „222“ und sendet aktuelle Temperatur, und Status von das RFID-reader. Paket sieht folgendes aus: 222,Temperatur,0/1;. Falls es wurde falsche Schlüssel benutzt, wird als drittes wert „1“ gesendet.

Das war eine Kurze Beschreibung und Erläuterung von Code für Arduino Uno, jetzt kommen wir zum Esp8266. Gleich als erstes werden folgende Werte definiert: SSID, Passwort, API-Token von Telegram Bot. Und Chat ID. Chat ID ist insbesondere wichtig, weil wenn der fehlt, kann der Bot nicht alle Funktionen erledigen, eine davon ist das der dann nicht selber Nachrichten senden kann. Mit „selber“ ist gemeint, das wenn beispielsweise ein Sensor kritische Werte liest, kann der Bot nicht Nachricht senden, aber er kann trotzdem Text aus Message lesen und auf die in Chat antworten.

Da werden folgende Bibliotheken verwenden: Fastbot.h für Telegram Bot, CharPlot.h um ESP8266 Grafiken mit ASCII Zeichen kann, und LittleFS.h. LitteFS.h wurde benutzt, Weil es ist Sinnvoll Werte die Nutzer für Temperatur anstellt, irgendwo speichern, genau dass kann diese Bibliothek ermöglichen. Die nutzt Flash Speicher die extra dafür da ist, falls man nutzt das nicht kann man das ausschalten, und dann hat man paar Megabyte mehr für Sketch.

Als Nächstes ist Initialisation von Bot, Als Parameter kriegt es API-Token.

Danach wird bool restart angelegt, String temperaturFile wo wird Temperatur gespeichert, und String lichtSt wo Status von Licht gespeichert wird. Array mit Größe 40, da werden werte für Grafik gespeichert. Es ist auf 40 gesetzt, weil es ist in die Lib-Doku vorgeschrieben, es wurde probiert was passiert, wenn die bisschen größer wird. Als Ergebnis kam raus unlesbare Grafik dann man kann nicht lesen.

IN setup() als erstes wird Funktion connectWiFi() aufgerufen, in die wird ESP zum Internet verbunden, wenn die länger als 15 Sekunden verbindet, wird die restartet. Danach wird Funktion geschrieben die Nachrichten bearbeiten wird. Es sieht so aus: bot.attach(newMsg);, und bedeutet dass in Funktion newMsg wird Strukt der mit Nachrichten arbeitet. Als Nächstes sendet Bot Nachricht das es Verbunden ist. In Code wird LED_BUILDIN initialisiert, Das ist LED der sich auf die ESP befindet. Auf Arduino Uno das ist Pin 13. Das wurde gebraucht, um die Verbindung und Arbeitsfähigkeit zu überprüfen. Danach wird ein Menu erstellt, das ist ein Panel mit Tasten über das Nutzer mit ESP kommuniziert. Danach wird File System gestartet, und Nachricht gesendet, in denen steht das File System wurde erfolgreich gestartet. Anschließend wird Funktion sendData() aufgerufen.

In sendData() wird als erstes Zeit von Telegram Server bekommen, und in struct gespeichert. Ziffer 1 in die Klammer steht für die Zeitzone. Als erste Wert wird Schlüssel „111“ gesendet. Danach kommen Stunden und Minuten aus struct. Und anschließend die Temperatur gesendet, die Nutzer mithilfe von taste „Edit Temperatur“ eingeben kann.

Bei diese Telegram Bot gibt mehrere Kommanden, die vorher schon erwähnt wurden. Die alle werden in newMsg() bearbeitet. Die Funktion newMsg() wird in loop() aufgerufen.

Es gebt folgende Kommanden:

State: Zeigt Status, Es besteht aus Aktuelle Temperatur, ob Licht an oder aus ist, und Status von BUILDIN_LED

Licht an oder aus machen Gelbes LED an oder aus. Bei dieser Funktion wird File System benutzt, es sieht so aus:

```
File temp = LittleFS.open(lichtSt, "a");
    temp.truncate(0);
    temp.print("1,");
    bot.sendMessage("Licht wird angemacht", CHAT_ID);


    sendData();
    temp.close();
```

da wird als erstes File geöffnet, truncate(0) löscht alle aus Datei aus,н

in die Datei wird „1," geschrieben und eine Nachricht gesendet, am Ende wird File geschlossen. Das Gleiche arbeitet bei „Licht aus" aber es wird anstatt „1," Satz aus "0," gespeichert.

Nächste Kommand ist „Temperatur zeigen" da wird Größe 40 erstellt und mit eine Zeile Grafik gezeichnet.

```
bot.sendMessage(CharPlot<COLON_X2>(arr, sz, 10, 0), CHAT_ID);
```

Taste Edit Temperatur wird eine inlineMenu aufgerufen. Es ist eine Nachricht mit zwei taste, bei drück auf eine davon wird Kommand aufgerufen die der Name das Taste haben. Bei „plus 1" wird temperaturSD um 1 erhört und in File gespeichert. Anschließend wird sendDate() aufgerufen.

Letzte Taste ist restart. Die setzt Variable restart auf „1", danach wird es in loop() geprüft, und wenn es „1" ist bot.tick auf Manual gesetzt und restartet. Es kann eine logische fragen kommen, warum das steht, nicht in die Komande „restart", Antwort ist relativ einfach. Wenn man das in die Kommand setzt, wird die dann unendlich restarten, weil letzte gesendete Nachricht von Nutzer ist „restart". Sowas muss man nur bei ESP.restart(); achten.

readDaten() ist eine letzte Funktion die Daten von Serial liest. Die funktioniert genau so wie bei Arduino Uno.

Probleme beim Projekt

Bei Projekt gab viele große und kleine Problemen. Erste große Problem war das Arduino gleichzeitig nur ein eine Aufgabe erledigen kann, das hat Bibliothek GyverOS gelöscht.

Es gab auch kleinere Probleme wie zum Beispiel DHT11 Sensor der immer wert „100.00" rausgelesen hat, das Problem wurde gelöscht durch Ersetzung mit anderem Sensor.

Es gab ein sehr großes Problem bei Funktion Klingel. Am ende komplett gelöscht aus Projekt. Problem lag daran, dass entweder hat gespammt in Serial oder gar nicht funktioniert. Es zwei Möglichkeiten die zu Problem zu löschen die nicht zum Projekt passen. Este Möglichkeit ist das delay auf 1000 setzen, aber danach macht das Programm ganze Sekunde gar nichts. Oder die Taste zum ESP anschließen, aber es wurde geplant das die ESP wird nur als Schnittstelle arbeiten.

Quellen

https://github.com/GyverLibs/FastBot/blob/main/README_EN.md

https://github.com/littlefs-project/littlefs

https://github.com/GyverLibs/GyverOS/blob/main/README_EN.md

https://github.com/GyverLibs/CharDisplay/blob/main/README_EN.md

https://github.com/song940/RFID-RC522/tree/master