



DIPLOMA THESIS

Smart Home Controlpanel

Add subtitle

Performed in 2025/26 by:

Fabian Schätzschock
Richard Krammer

Supervisors:

Supervisor 1
Supervisor 2

St. Pölten, on 04.04.2025

Submission Statement:
Date:

Supervisors:

Affidavit

The undersigned candidates have chosen to prepare a diploma thesis with the following task description in accordance with the Schulunterrichtsgesetz (School Education Act) § 34 Abs. 3 Z 1 and § 37 Abs. 2 Z 2 [1], in conjunction with the provisions of the Prüfungsordnung BMHS (Examination Regulations for Vocational Middle and Higher Schools), Federal Law Gazette II No. 177/2012 [2], as amended

Smart Home Controlpanel

Add subtitle

Individual tasks within the overall project:

- **Fabian Schätzschock:** Task 1
- **Richard Krammer:** Task 2

The candidates acknowledge that the diploma thesis must be worked on and completed independently and outside of class time, although class results may be incorporated if appropriately cited as such.

The complete diploma thesis must be submitted digitally and in two printed copies to the supervising teacher no later than **04.04.2025**.

The candidates further acknowledge that cancellation of the diploma thesis is not possible.

Fabian Schätzschock

Richard Krammer

Contents

1	Einführung	1
1.1	Software	1
1.2	Hardware	1
2	Anforderungen	3
2.1	Software	3
2.1.1	MQTT	3
2.1.2	ESP-NOW	3
2.1.3	Display	3
2.1.4	UI-Library	3
2.1.5	Display	4
2.2	Hardware	4
2.2.1	MCU	4
2.2.2	Eingabemöglichkeit	4
2.2.3	Display	4
2.2.4	ARGB	4
3	Software	5
3.1	Projektstruktur	5
3.2	ESP32-S3 Konfiguration	5
3.3	User Interface	5
3.3.1	SquarelineStudio	5
3.3.2	Touch Control	6
3.4	Kommunikation	9
3.4.1	MQTT	9
3.4.2	ESP-NOW	9
4	Ergebnisse	11
5	Time Tracking	13
5.1	Name 1	13
	Bibliography	15

1 Einführung

1.1 Software

Das Control-Panel soll mit dem Netzwerk eines Smarthomes verbunden werden und in der Lage sein, über den im Smarthome bereits integrierten MQTT-Broker mit integrierten Geräten zu kommunizieren und den Status dieser Geräte abzufragen und anzupassen.

1.2 Hardware

2 Anforderungen

2.1 Software

Zur effizienten Realisierung des Panels sind zwei Hauptfunktionalitäten zu implementieren.

2.1.1 MQTT

MQTT ist ein Protokoll, welches über TCP/IP arbeitet und besonders für IoT Anwendungen geeignet ist. Der Raspberry Pi fungiert als Broker, welcher die Kommunikation zwischen den verschiedenen Geräten ermöglicht.

2.1.2 ESP-NOW

Eine mögliche Alternative zu MQTT ist ESP-NOW. Hierbei handelt es sich um ein Protokoll, welches speziell für die Kommunikation zwischen ESP-Chips entwickelt wurde. Da die Chips selber als Access-Points fungieren, ist kein bestehendes Netzwerk notwendig. Unter idealen Bedingungen ist eine Reichweite von bis zu 200m möglich.

2.1.3 Display

Bei der Umsetzung des Displays sind zwei Funktionalitäten die es zu implementieren gilt. Zum einen die Anzeige von Informationen und zum anderen die Interaktion mit dem User Interface.

2.1.4 UI-Library

Für die Ansteuerung des Displays wurde die "TFT_eSPI" Library von "Bodmer" verwendet. Diese Library bietet Support für den am Display verwendeten Controller.

Input

Die Library bietet die Möglichkeit, Touch-Events zu registrieren und zu verarbeiten. Da das registrieren des Touches nur über SPI möglich ist, wird die

"XPT2046_Touchscreen" Library verwendet. Der Output soll anschließend an die "TFT_eSPI" Library weitergeleitet werden.

2.1.5 Display

Bei der Umsetzung des Displays sind zwei Funktionalitäten die es zu implementieren gilt. Zum einen die Anzeige von Informationen und zum anderen die Interaktion mit dem User Interface.

2.2 Hardware

2.2.1 MCU

Es wurde sich für einen ESP-32 S3 als Microcontroller entschieden. Dieser ist mit diverser Funktion (Bluetooth, WLAN, ESPNOW ...) sowie genügend Port-Pins ausgestattet um die Information aus dem Server auf den Bildschirm anzuzeigen.

2.2.2 Eingabemöglichkeit

Um mit dem Produkt zu interagieren wurde versucht das Resistive Touch-Panel, welches bereits auf dem Display vormontiert war, zu verwenden. Durch Schwierigkeiten bei der Implementierung in der Software, wurde sich jetzt für einen Rotary-Encoder als Input entschieden. Es wird dann durch Drehen und Drücken durch das Menü navigiert.

2.2.3 Display

Das Display ist mit dem SDD1963 Controller bestückt und kann über einen 16-Bit oder 8-Bit Bus angesteuert werden, letzteres wurde in der im Projekt implementiert. Das Display wurde für seine Größe (7 Zoll) und wegen der Farbdarstellungen ausgewählt.

2.2.4 ARGB

Für eine intuitive Benutzung wird zusätzlich ein Piezo für akustische Signale sowie ARGB LEDs für optische Signale eingeplant. Diese sollen den Anwender durch Licht/Audio einen groben Überblick geben, auch wenn das Control-Panel weiter entfernt ist.

3 Software

3.1 Projektstruktur

Die grundlegende Struktur des Projekts folgt der von PlatformIO normalisierten Projektstruktur. Die Funktionen werden nach ihrer Verwendung in verschiedene Dateien in der *FaRiLib* Library aufgeteilt, diese ist in ein *src* und *include* Directory aufgegliedert.

3.2 ESP32-S3 Konfiguration

Die Eigenschaften des ESP32-S3 werden in dem *platformio.ini* File konfiguriert. Als Basis der Konfiguration dient das *esp32-s3-devkitc-1.json* File, welches die Basiskonfiguration für ein ESP32-S3 Devkit bereitstellt und standardmäßig von PlatformIO verfügbar ist.

Nun müssen folgende Flags im *platformio.ini* File überschrieben werden:

```
board_upload.flash_size = 16MB
board_build.partitions = default_16MB.csv
```

Nun sollte der Upload eines Programms auf den ESP-Chip möglich sein.

3.3 User Interface

3.3.1 SquarelineStudio

SquarelineStudio ist eine Software, die es ermöglicht, ein User Interface mithilfe eines Drag-and-Drop-Editors zu erstellen und in C-Code zu exportieren. Dieser Code kann dann zusammen mit den beiden Libraries *TFT_eSPI* und *lvgl* in PlatformIO integriert werden.

TFT_eSPI

TFT_eSPI ist eine Library für Grafik und Fonts auf einem TFT-Display. Sie ist mit vielen verschiedenen Controllern kompatibel und bietet viele Funktionen, um verschiedene TFT-Display anzusteuern. Sie ist eine Hälfte des Grundgerüsts für die Darstellung von Grafiken in SquarelineStudio.

lvgl

lvgl ist eine Library für die Darstellung von flexiblen Grafiken auf vielen Plattformen, darunter auch dem Arduino Framework. Zusammen mit TFT_eSPI bildet sie das Grundgerüst für die Darstellung von Grafiken in SquarelineStudio.

3.3.2 Touch Control

Die *TFT_eSPI* Library bietet auch die Möglichkeit, Touch-Events zu registrieren und für das *SquarelineStudio* User Interface zu verarbeiten. Diese Funktion ist allerdings nur für über SPI angesteuerte Displays verfügbar.

XPT2046_Touchscreen Library

Um den Touch seriell einzulesen, wird die *XPT2046_Touchscreen* Library verwendet.

```
#define CS_PIN 10

XPT2046_Touchscreen ts(CS_PIN);
TS_Point p;
```

Touch mit tft_eSPI koppeln

Um die Touch-Events an tft_eSPI weiterzuleiten, muss die library grundlegend umgeschrieben werden. zunächst muss das User_Setup.h folgender Maßen bearbeitet werden:

```
//Einkommentieren
#define TOUCH_CS 10
```

Ebenso muss im "tft_eSPI.h" folgendes editiert werden:

```
#ifdef TOUCH_CS
//CHANGE THIS LINE
#if 0
    #if !defined(DISABLE_ALL_LIBRARY_WARNINGS)
        #error >>>----->> ...
    #endif
#else
    #include "Extensions/Touch.h"
#endif
#else
    #if !defined(DISABLE_ALL_LIBRARY_WARNINGS)
        #warning >>>----->> ...
    #endif
#endif
```

Nun kann der Touch in der `my_touchpad_read()` Funktion der `tft_eSPI` Library übergeben werden.

```
void my_touchpad_read
(lv_indev_drv_t*indev_driver, lv_indev_data_t*data)
{
    uint16_t touchX = 0, touchY = 0;

    if( !ts.touched() || p.z < 1950 )
    {
        data->state = LV_INDEV_STATE_REL;
    }
    else
    {
        data->state = LV_INDEV_STATE_PR;

        p = ts.getPoint();

        /*Set the coordinates*/
        data->point.x = p.x;
        data->point.y = p.y;

        Serial.print( "Data_x_" );
        Serial.println(data->point.x-700);

        Serial.print( "Data_y_" );
        Serial.println(data->point.y);
    }
}
```

Touch wurde nicht implementiert

Die `tft_eSPI` Library verhindert durch die in section 3.3.2 beschriebenen `#if` Makros, die Implementierung des Touches. Trotz mehrerer Versuche, die Library zu modifizieren, wirft die library immer wieder `include-Fehler` in dem dann inkludierem `Touch.h` file.

3.4 Kommunikation

3.4.1 MQTT

Um eine Kommunikation zu ermöglichen, ist eine MQTT-Library notwendig. Hierfür wurde "PubSubClient" von "knolleary" verwendet. Hierbei handelt es sich um eine simple Library, die nur das Nötigste implementiert, um einen überschaubaren Overhead zu gewährleisten.

3.4.2 ESP-NOW

4 Ergebnisse

5 Time Tracking

5.1 Name 1

2024		
Week	Task Description	Hours
36	Preparation	12

2025		
Week	Task Description	Hours
35	Polishing	12

Total		24
-------	--	----

Bibliography

- [1] *Bundesgesetz über die Ordnung von Unterricht und Erziehung in den im Schulorganisationsgesetz geregelten Schulen.* 1986. URL: <https://www.ris.bka.gv.at/GeltendeFassung.wxe?Abfrage=Bundesnormen&Gesetzesnummer=10009600>.
- [2] *Prüfungsordnung BMHS, Bildungsanstalten.* 2012. URL: <https://www.ris.bka.gv.at/GeltendeFassung.wxe?Abfrage=Bundesnormen&Gesetzesnummer=20007845>.