Türklingel

Übersicht

Im Hackerspace ist am Eingang Bornstr. 14-15 eine Türklingel mit Sprechanlage (Türsprechstelle) installiert. Das Haustelefon befindet sich im Workshopraum.

Es wurde eine Möglichkeit gefunden, das Klingelsignal abzugreifen und in die Räume Bornstr. 16-17 zu verlängern (E- und Kreativwerkstatt).

Optional:

Es könnte zu jeder zeit an sender und oder Empfängern eine Li-ion Akku (via JST) angeschlossen werden und somit auch mobil betrieben werden z.b bei Veranstaltungen die im Hinnenhof stattfinden um kein Klingeln zu verpassen.

Achtung bei Powerbanks!

Da sich die meisten Powerbanks bei geringer Last ausschalten, könnte es sein, dass der Empfänger bei mobilen betrieb mit Powerbanks die Funktion nach kurzer zeit Einstellen (https://apfelhirn.de/automatische-abschaltung-von-powerbanks-als-arduino-stromversorgung-bei-geringer-grundlast-verhindern/).

Sprechanlage

Die Sprechanlage nutzt ein 2-Draht-Bussystem vom Typ STR QwikBus, welches die Türsprechstelle mit den Haustelefonen verbindet. Daher ist es nicht möglich, das Klingelsignal direkt abzugreifen, da es digital über den Bus übertragen wird.

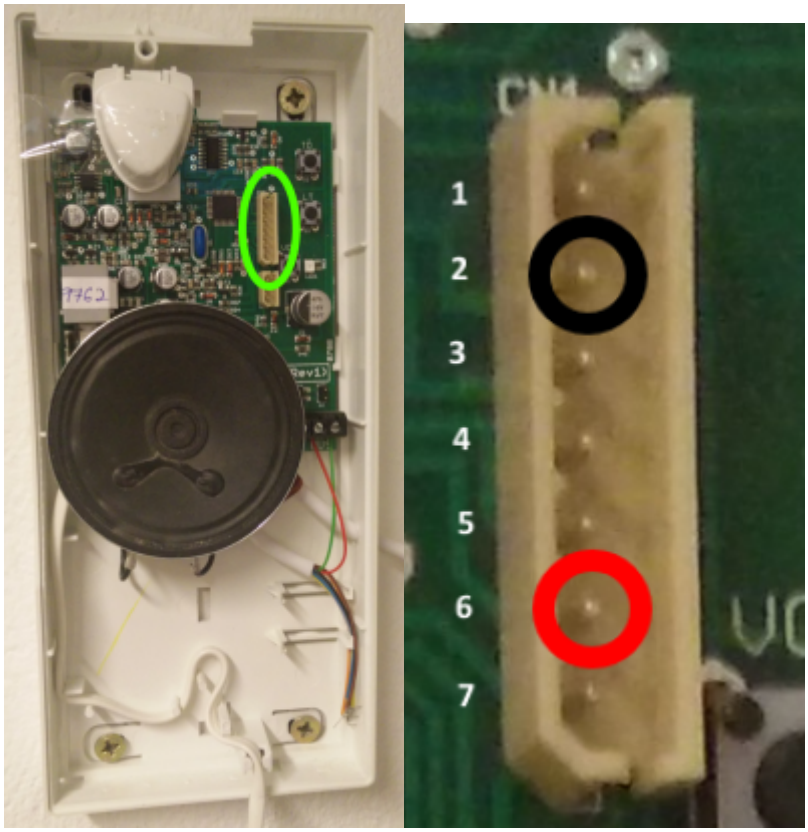
Das Haustelefon ist vom Typ STR HT 3033.

Das Steuergerät ist ein SP 333 (Doku), welches im Hauskeller im Sicherungsschrank neben dem Netzteil NH 333 installiert ist.

Pinout

Öffnet man das Haustelefon über die einzig vorhandene Gehäuseschraube (die obere Gehäusehälfte lässt sich dann herunterklappen), so kommen innen zwei Steckverbinder zum Vorschein.

Die Belegung lässt sich durch ein Foto einer Relaisbox AM333, die man hinzukaufen kann, erahnen. Ein Multimeter half bei der weiteren Analyse. Auf dem zweiten Pin von oben liegt Masse, auf dem zweiten Pin von unten wird das Klingelsignal ausgegeben (5 V, wenn ich mich recht erinnere).



Mechanisch konnten die passenden Steckerweibchen nicht ermittelt werden. Daher greifen wir das Signal über zwei IC-Beinchen-Einsätze gedrehter IC-Sockel ab, die in den Steckverbindern verlässlich halten.

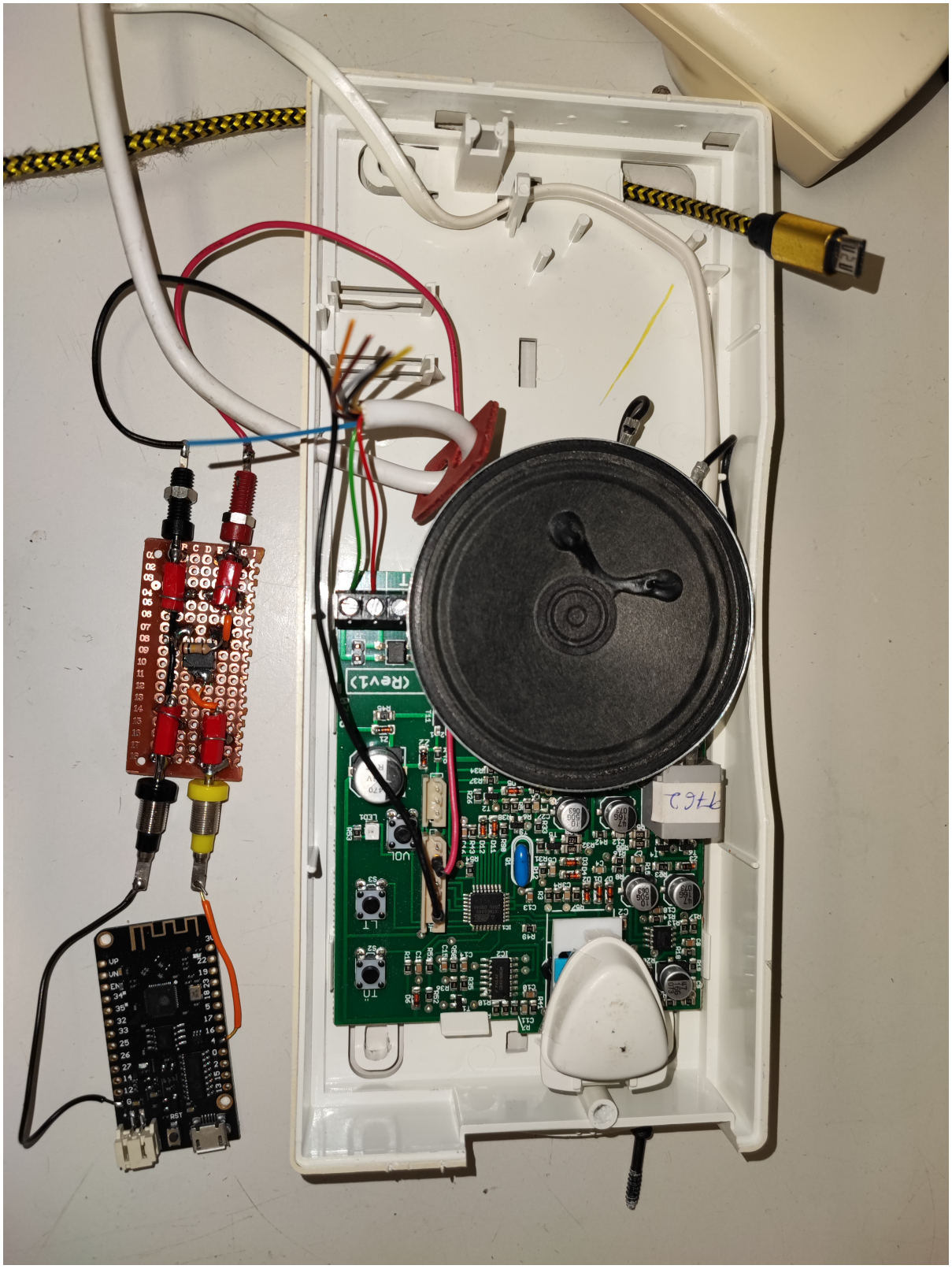
Funk-Türklingel

Da uns keine Leitung zwischen den Räumen Bornstr. 14-15 sowie 16-17 zur Verfügung steht, mussten wir auf eine Funk-Türklingel mit zwei Empfängern zurückgreifen. Der Sender sowie die 2 Empfänger werden mit einem Standard 5 Volt USB Netzteil versorgt.

Wenn geklingelt wird, zieht ein n channel mosfet den Pin 4 des esp32 auf ground. Wenn dieses > 1000 ms auf Ground bleibt, wurde geklingelt und ein ESPNOW signal zu den Empfängern geschickt.

Sender in Haustelefon einbauen

Der Sender ist ein ESP32 Lolin Lite welcher auf ein Trigger auf Pin 4 wartet (als sensor wird ein n chanel mosfet benutzt. (Bild: Patine mitte links)



Sourcecode

Sender:

```
#include <WiFi.h>
#include <esp_wifi.h>
#include <esp_now.h>
```

Set your new MAC Address
uint8_t newMACAddress[] = {0x30, 0xAF, 0xA0, 0x05, 0x1D, 0x63}; REPLACE WITH YOUR ESP RECEIVER'S MAC ADDRESS
uint8_t broadcastAddress1[] = {0x30, 0xAF, 0xA1, 0x05, 0x1D, 0x63};
uint8_t broadcastAddress2[] = {0x30, 0xAF, 0xA2, 0x05, 0x1D, 0x63};

typedef struct bell_struct {
uint64_t trigger;
} bell_struct;

bell_struct bell;

esp_now_peer_info_t peerInfo;

callback when data is sent
*void OnDataSent(const uint8_t *mac_addr, esp_now_send_status_t status) {*
char macStr[18];
Serial.print(„Packet to: “);
Copies the sender mac address to a string
snprintf(macStr, sizeof(macStr), „%02x:%02x:%02x:%02x:%02x:%02x“,

```
       mac_addr[0], mac_addr[1], mac_addr[2], mac_addr[3], mac_addr[4],
mac_addr[5]);\\
```

Serial.print(macStr);
Serial.print(„ send status:\t“);
Serial.println(status == ESP_NOW_SEND_SUCCESS ? „Delivery Success“ : „Delivery Fail“);
}

void setup() {
Serial.begin(115200);

WiFi.mode(WIFI_STA);

Set the ESP32 Board MAC Address.
esp_wifi_set_mac(WIFI_IF_STA, &newMACAddress[0]); Print the new MAC Address.
Serial.print(„[NEW] ESP32 Board MAC Address: “);
Serial.println(WiFi.macAddress());

if (esp_now_init() != ESP_OK) {

```
Serial.println("Error initializing ESP-NOW");\\
ESP.restart();\\
```

}

esp_now_register_send_cb(OnDataSent);

register peer
peerInfo.channel = 0;
peerInfo.encrypt = false;

register first peer
memcpy(peerInfo.peer_addr, broadcastAddress1, 6);
if (esp_now_add_peer(&peerInfo) != ESP_OK) {

```
Serial.println("Failed to add peer");\\
return;\\
```

}

register second peer
memcpy(peerInfo.peer_addr, broadcastAddress2, 6);
if (esp_now_add_peer(&peerInfo) != ESP_OK) {
Serial.println(„Failed to add peer“);
return;
} Set pin 4 as an interrupt
pinMode(4, INPUT_PULLUP);
attachInterrupt(digitalPinToInterrupt(4), triggerbell, FALLING);
}

void triggerbell() {
bell.trigger = 8562190430;

esp_err_t result = esp_now_send(0, (uint8_t *) &bell, sizeof(bell_struct));

if (result == ESP_OK) {

```
Serial.println("Sent with success");\\
```

} else {

```
Serial.println("Error sending the data");\\
```

}

Serial.print(„[NEW] ESP32 Board MAC Address: “);
Serial.println(WiFi.macAddress());

}

void loop() {

}

Empfänger1

#include <WiFi.h>
#include <esp_wifi.h>
#include <esp_now.h>

Set MAC Address this device (bell receiver 1)
uint8_t newMACAddress[] = {0x30, 0xAF, 0xA1, 0x05, 0x1D, 0x63}; uint8_t addr[] = {0x30, 0xAF, 0xA0, 0x05, 0x1D, 0x63}; unsigned long lastPrintoutTime = 0;

bool isTriggered = false;
unsigned long lastTriggeredTime = 0; typedef struct bell_struct {
uint64_t trigger;
} bell_struct; Create a struct_message called myData
bell_struct myData;

void OnDataRecv(const uint8_t * mac, const uint8_t *incomingData, int len) {
memcpy(&myData, incomingData, sizeof(myData));
Serial.print(„Bytes received: “);
Serial.println(len);
Serial.print(„received command:“);
Serial.println(myData.trigger);
if (memcmp(mac, addr, 6) == 0) { *Check if expected Mac Addr*
Serial.println(„klingel device received!“);
if (myData.trigger == 8562190430) {
Serial.println(„valid Trigger code received!“);
triggerbell();
}
}
} void setup() { Serial.begin(115200); Serial.println(); WiFi.mode(WIFI_STA); Serial.print(„[OLD] ESP32 Board MAC Address: “);
Serial.println(WiFi.macAddress()); ESP32 Board add-on before version < 1.0.5
esp_wifi_set_mac(ESP_IF_WIFI_STA, &newMACAddress[0]); ESP32 Board add-on after version > 1.0.5
esp_wifi_set_mac(WIFI_IF_STA, &newMACAddress[0]);

Serial.print(„[NEW] ESP32 Board MAC Address: “);
Serial.println(WiFi.macAddress());

pinMode(LED_BUILTIN, OUTPUT);
pinMode(4, OUTPUT);

digitalWrite(LED_BUILTIN, LOW);
digitalWrite(4, HIGH); *trigger bell*
delay(100);
digitalWrite(4, LOW);
digitalWrite(LED_BUILTIN, HIGH);
delay(100); Set device as a Wi-Fi Station
WiFi.mode(WIFI_STA);

Init ESP-NOW
if (esp_now_init() != ESP_OK) {
Serial.println(„Error initializing ESP-NOW“);
ESP.restart();
} Once ESPNow is successfully Init, we will register for recv CB to
get recv packer info
esp_now_register_recv_cb(OnDataRecv);
}
void triggerbell() {
if (!isTriggered || (millis() - lastTriggeredTime > 10000)) {
isTriggered = true;
lastTriggeredTime = millis(); digitalWrite(LED_BUILTIN, LOW);
digitalWrite(4, HIGH); trigger bell

```
delay(100);\\
digitalWrite(4, LOW);\\
digitalWrite(LED_BUILTIN, HIGH);\\
Serial.println("bell triggered");\\
```

} }

void loop() {
Print out the `isTriggered` variable every second
if (millis() - lastPrintoutTime > 1000) {
Serial.print(„isTriggered: “);
Serial.println(isTriggered);
lastPrintoutTime = millis();
}

Reset the `isTriggered` variable after 5 seconds
if (millis() - lastTriggeredTime > 5000) {

```
isTriggered = false;\\
```

}

}

Empfänger2

#include <WiFi.h>
#include <esp_wifi.h>
#include <esp_now.h>

Set MAC Address this device (bell receiver 2)
uint8_t newMACAddress[] = {0x30, 0xAF, 0xA2, 0x05, 0x1D, 0x63}; uint8_t addr[] = {0x30, 0xAF, 0xA0, 0x05, 0x1D, 0x63}; unsigned long lastPrintoutTime = 0;
bool isTriggered = false;
unsigned long lastTriggeredTime = 0; typedef struct bell_struct {
uint64_t trigger;
} bell_struct; Create a struct_message called myData
bell_struct myData;

void OnDataRecv(const uint8_t * mac, const uint8_t *incomingData, int len) {
memcpy(&myData, incomingData, sizeof(myData));
Serial.print(„Bytes received: “);
Serial.println(len);
Serial.print(„received command:“);
Serial.println(myData.trigger);
if (memcmp(mac, addr, 6) == 0) { *Check if expected Mac Addr*
Serial.println(„klingel device received!“);
if (myData.trigger == 8562190430) {
Serial.println(„valid Trigger code received!“);
triggerbell();
}
}

} void setup() { Serial.begin(115200); Serial.println(); WiFi.mode(WIFI_STA); Serial.print(„[OLD] ESP32 Board MAC Address: “);
Serial.println(WiFi.macAddress()); ESP32 Board add-on before version < 1.0.5
esp_wifi_set_mac(ESP_IF_WIFI_STA, &newMACAddress[0]); ESP32 Board add-on after version > 1.0.5
esp_wifi_set_mac(WIFI_IF_STA, &newMACAddress[0]);

Serial.print(„[NEW] ESP32 Board MAC Address: “);
Serial.println(WiFi.macAddress());

pinMode(LED_BUILTIN, OUTPUT);
pinMode(4, OUTPUT);

digitalWrite(LED_BUILTIN, LOW);
digitalWrite(4, HIGH); *trigger bell*
delay(100);
digitalWrite(4, LOW);
digitalWrite(LED_BUILTIN, HIGH);
delay(100); Set device as a Wi-Fi Station
WiFi.mode(WIFI_STA);

Init ESP-NOW
if (esp_now_init() != ESP_OK) {
Serial.println(„Error initializing ESP-NOW“);
ESP.restart();
} Once ESPNow is successfully Init, we will register for recv CB to
get recv packer info
esp_now_register_recv_cb(OnDataRecv);
}
void triggerbell() {
if (!isTriggered || (millis() - lastTriggeredTime > 10000)) {
isTriggered = true;
lastTriggeredTime = millis(); digitalWrite(LED_BUILTIN, LOW);
digitalWrite(4, HIGH); trigger bell

```
delay(100);\\
digitalWrite(4, LOW);\\
digitalWrite(LED_BUILTIN, HIGH);\\
Serial.println("bell triggered");\\
```

} }

void loop() {
Print out the `isTriggered` variable every second
if (millis() - lastPrintoutTime > 1000) {
Serial.print(„isTriggered: “);
Serial.println(isTriggered);
lastPrintoutTime = millis();
}

Reset the `isTriggered` variable after 5 seconds
if (millis() - lastTriggeredTime > 5000) {

```
isTriggered = false;\\

}

}
```