



Smart Home

Mach dein Zuhause ein klein wenig sicherer!



Unsere Vision

- Sichere und zukunftsfähige Sicherheitslösung für Smart Home
 - Kein “discontinued product” ohne Updates
 - Keine abgeschaltete Cloud
 - Kein Zwang Version n+1 zu kaufen, weil ältere Versionen nicht mehr unterstützt werden

Umweltschonende Wiederverwendung funktionierender Hardware



Unsere Vision

- Einbruchserkennung mit State-of-the-Art Verschlüsselung
 - Funkstandard 802.11 mit WPA2
 - SSL verschlüsselte Kommunikation mit der Smart Home Zentrale
 - Push und Email-Notifications bei Alarm
 - Erkennung von Störungen im Funkkanal
 - Ausfallerkennung von Knoten

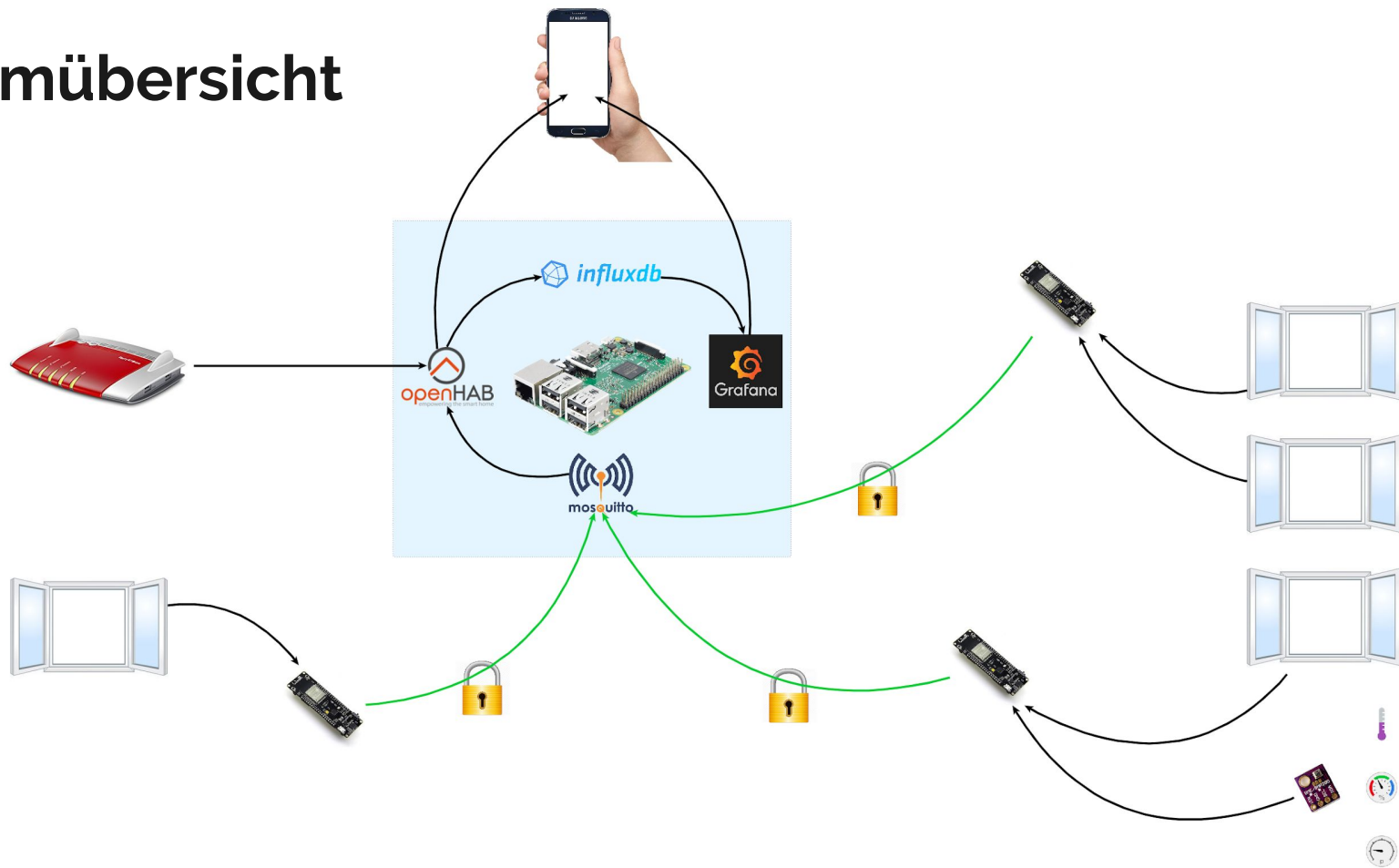


Unsere Vision

- Selbstgehostete Smart Home Lösung
 - Opensource
 - Aktualisierung/ Modifizierungen möglich
 - Austausch einzelner Komponenten möglich
 - Erweiterbarkeit
 - Weitere Sensoren (Umweltsensoren)
 - Einbindung weiterer Smart Home Geräte

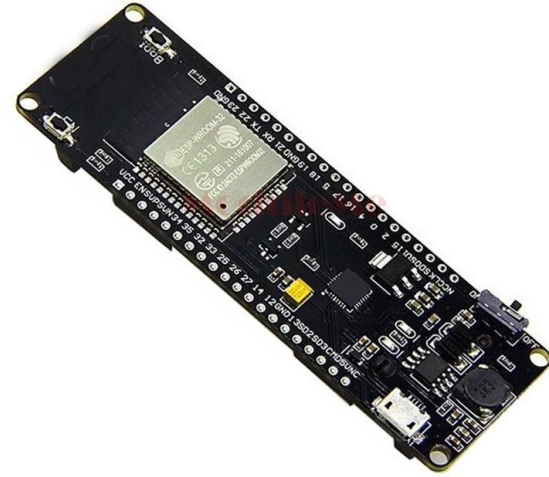
Aufbau des Projektes

Systemübersicht



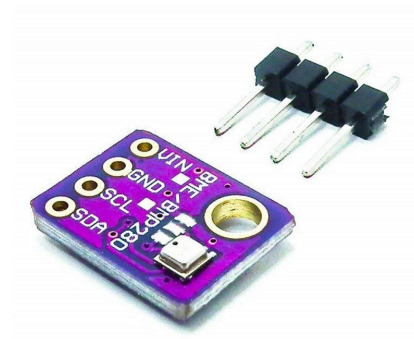
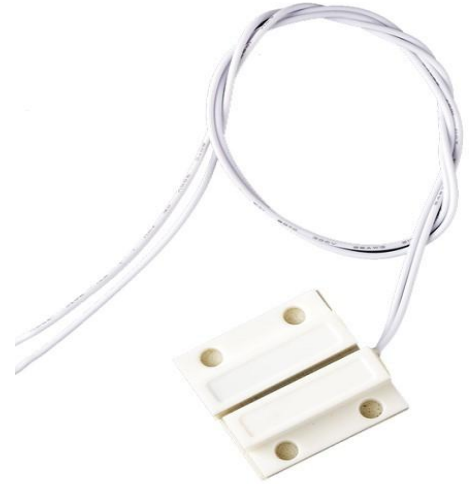
Hardware

- ESP32
 - 4 MB Flash
 - Dual Core
 - Ladetechnik und 18650 Akku Anschluss
 - Eingebautes Funkmodule (WLAN + Bluetooth)

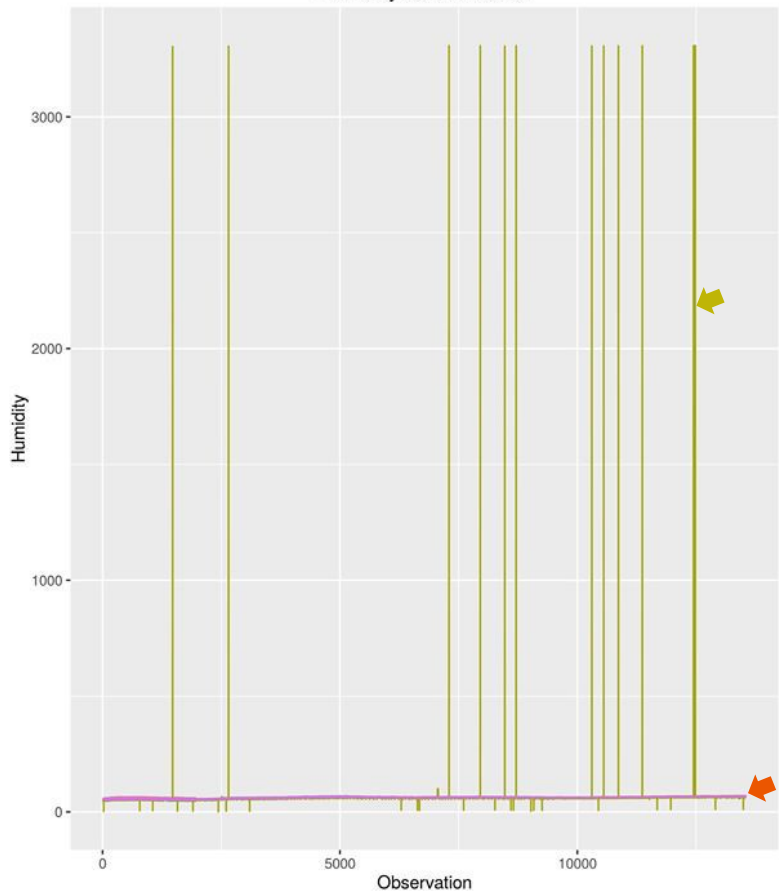


Sensoren

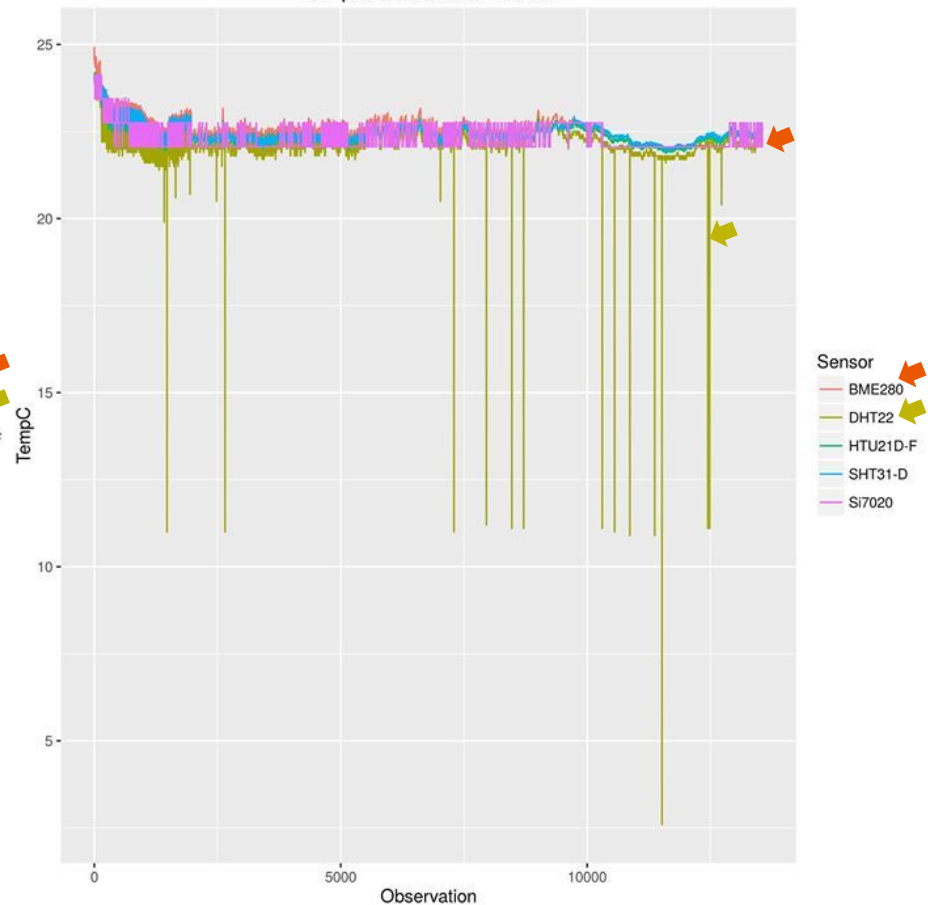
- Einfache Magnetsensoren für Fenster und Türen
- BME280/ BME680 für Umweltmessungen
 - Bessere Qualität als DHT 11/22
 - Zusätzlich Luftdruck / Gas



Humidity Observations



Temperature Observations





Gesamtkosten für ein System-Paket

- Raspberry Pi 3 (inkl. WLAN + Bluetooth + Gehäuse, SD-Karte, Netzteil)
 - ~60,00 €
- Wemos ESP32 Board (inkl. Li-Ion Batterie) - 4 Stück
 - ~60,00 €
- Sensoren (Magnetsensoren (8), BME280 (4) oder BME680 (4))
 - ~30,00 €

Gesamt: ~150,00 €



openHAB (Open Home Automation Bus)



- Open Source Smart Home Lösung
- Auf Java Basis für alle gängigen Betriebssysteme
- Offizielles Projekt der Eclipse Foundation
- Große Community
- Gut erweiterbar



Automatisierte Einrichtung des Systems

- openHAB, influxDB, mosquitto, grafana, Zertifikate, ...
 - Übertragung sowie Modifizierung von Dateien auf dem Raspberry Pi mittels Fabric (SSH)
- ESP32 Toolchain
 - Umgebung und Abhängigkeiten für Mac und Linux Systeme eingerichtet



Kommunikation

- Mikrocontroller senden Daten über MQTT (Mosquitto)
- Smart Home Server betreibt auch MQTT-Broker
- Topics sind nach Räumen geordnet
 - “room/office/nodeID/WindowSensor1”
 - “room/livingroom/nodeID/humidity”

Sensorik

- Support von BME280/BME680
- Grafische Darstellung in Grafana
- Support von weiteren Sensoren möglich



Sicherheitsmerkmale



Angriffserkennung (Akustisches Signal)

- Anschluss eines Lautsprechers an der Smart Home Zentrale
- Bei Fensteröffnung während des aktiven Systems Abspielen eines Signaltons



Angriffserkennung (Störung Funkkanal)

- Smart Home Server (RaspberryPi 3 /3+) hat ein WLAN-Modul
- Das Modul überprüft per Cron die WLAN Qualität
- Alarm, sobald WLAN Qualität unter x % fällt
- Der Server für den tatsächlichen Betrieb über LAN angeschlossen (Erreichbarkeit von außen im Falle einer Funkstörung)



Lost Node Detection

- Verlorene Knoten senden keine Daten mehr
- Alle Knoten senden regelmäßigen Health Check, alle Checks werden persistiert (InfluxDB)
- Senden einer Mail/ Push Notification durch Grafana bei fehlenden Rückmeldungen
- MQTT Event → openHab → Influx → Grafana → Notification

Multihop Network

- **Idee:**
 - Weiterleitung der Signale über die ESP32 Boards zur Reichweitenverbesserung
- **Feature Abgelehnt:**
 - Widerspruch zu unseren Stromsparzielen
- **Alternative:**
 - WLAN Repeater im Heimnetzwerk zur Reichweitenverbesserung



Absicherung von Mosquitto

- Essentiell, da Transport der Nachrichten über Fensterstatus kritisch
- Einrichtung von TLS im Mosquitto MQTT Broker
 - Raspberry Pi dient als CA
- Sichere Authentifikation über Client Zertifikate
 - Ausstellung ebenfalls mit RaspberryPi CA
- Zusätzlich Benutzer und Passwort notwendig

Usability



OTA - Updates

- Erleichterte Updates bei bereits montierten Geräten
- Einführung von Versionsnamen und Versionscode zur Unterscheidung
- Minimalistischer HTTP-Server auf Raspberry Pi
- Suche nach Appversion mit höherem Versionscode
 - ggf. Download und Installation
- Gegenwärtig ungeschützt (bzw. lediglich über das Heimnetz)



Akkubetrieb

- Boards haben einen Akkuhalter (18650 -3400mAh)
- Ladevorgang über Board ist möglich
- Akkulebenszeit
 - Bei Aufwecken alle 15 Minuten : ~53 Tage*
 - Bei Aufwecken alle 60 Minuten : ~206 Tage*
 - Bei Aufwecken zweimal pro Tag: ~58 Monate*

* Selbstentladung nicht eingerechnet



Stromsparmaßnahmen

- Deep Sleep führt zu Reset des ESP32
 - Irrelevant für unseren Anwendungsfall (Light Sleep nahezu gleich schnell)
- Eingeschränkte GPIO Interrupts im Schlafmodus
 - External wakeup (ext0): **nur ein GPIO** konfigurierbar
 - External wakeup (ext1): Mehrere GPIO möglich, allerdings entweder “ANY_HIGH” oder “**ALL_LOW**”
 - maximal 2 Fenstersensoren pro Board
- Zusätzlich Timer zum Aufwecken

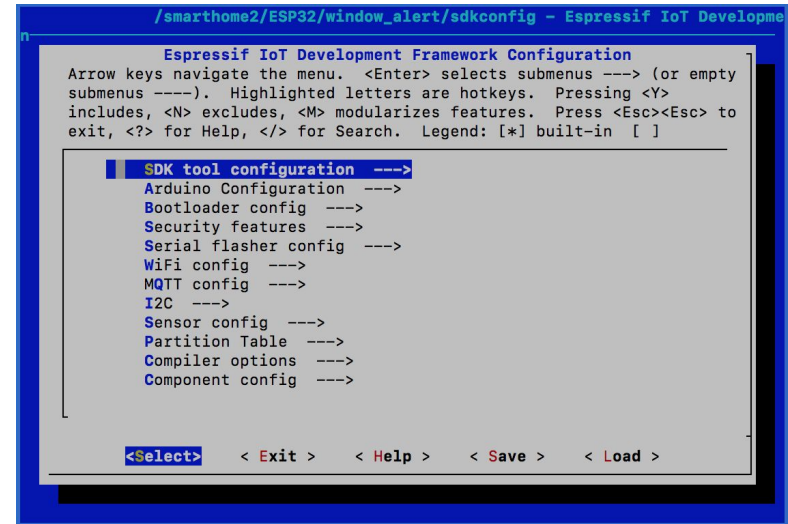


Convenience Features

- Automatische Aktivierung des Alarms
 - Momentan nur mit FritzBox
 - Sollten alle spezifizierten WLAN-Geräte offline sein (aus dem WLAN), wird das System aktiviert

Einfache und schnelle Einrichtung

- Sensoren an die Pins stecken
- Einstellungen im Menu
- USB Flash
- Montage
- (Debugging mit USB möglich)
- Fertig



The screenshot shows a terminal window titled "/smarthome2/ESP32/window_alert/sdkconfig - Espressif IoT Developme". The main content is the "Espressif IoT Development Framework Configuration" menu. It includes instructions on how to navigate using arrow keys, select submenus with <Enter>, and use hotkeys. The menu items are: SDK tool configuration (highlighted), Arduino Configuration, Bootloader config, Security features, Serial flasher config, WiFi config, MQTT config, I2C, Sensor config, Partition Table, Compiler options, and Component config. At the bottom, there are navigation options: <Select>, < Exit >, < Help >, < Save >, and < Load >.

```
/smarthome2/ESP32/window_alert/sdkconfig - Espressif IoT Developme

Espressif IoT Development Framework Configuration
Arrow keys navigate the menu. <Enter> selects submenus ----> (or empty
submenus ----). Highlighted letters are hotkeys. Pressing <Y>
includes, <N> excludes, <M> modularizes features. Press <Esc><Esc> to
exit, <?> for Help, </> for Search. Legend: [*] built-in [ ]

[*] SDK tool configuration ---->
  Arduino Configuration ---->
  Bootloader config ---->
  Security features ---->
  Serial flasher config ---->
  WiFi config ---->
  MQTT config ---->
  I2C ---->
  Sensor config ---->
  Partition Table ---->
  Compiler options ---->
  Component config ---->

<Select>  < Exit >  < Help >  < Save >  < Load >
```



Live Demo



Ausblick

- Konfigurator erfasst noch nicht alle Punkte
 - Boards lassen sich nicht automatisiert “in Reihe” flashen
- Ausbau von Sicherheitsfunktionen
 - Bewegungsmelder
 - Kamera
 - Steuerung von Smart Home Geräten (Licht an/aus) bei Alarm
- Angepasstes 3D-Gehäuse für die Boards
- OTA: HTTPS + Client Authentifikation
- Secure Bootloader, Flash Encryption