

The background of the advertisement features a large, multi-paned window with white frames. The window is set against a bright blue sky with scattered white clouds. Below the sky, a lush green field of tall grass is visible. The window panes are arranged in a way that creates a sense of depth and perspective. On the left side, a dark grey rectangular box contains the text 'SmartWindow' and 'Das etwas smartere Fenster'. To the right of this box, several hexagonal shapes of varying sizes are overlaid on the image, each containing a different view of the landscape or sky, suggesting the window's ability to change its view or filter light. The overall aesthetic is clean, modern, and minimalist.

SmartWindow

Das etwas smartere Fenster

Übersicht

Wieso ein
Smartes
Fenster?

Was sind die
Besonderheiten
im Code ?



Was wurde
benutzt?

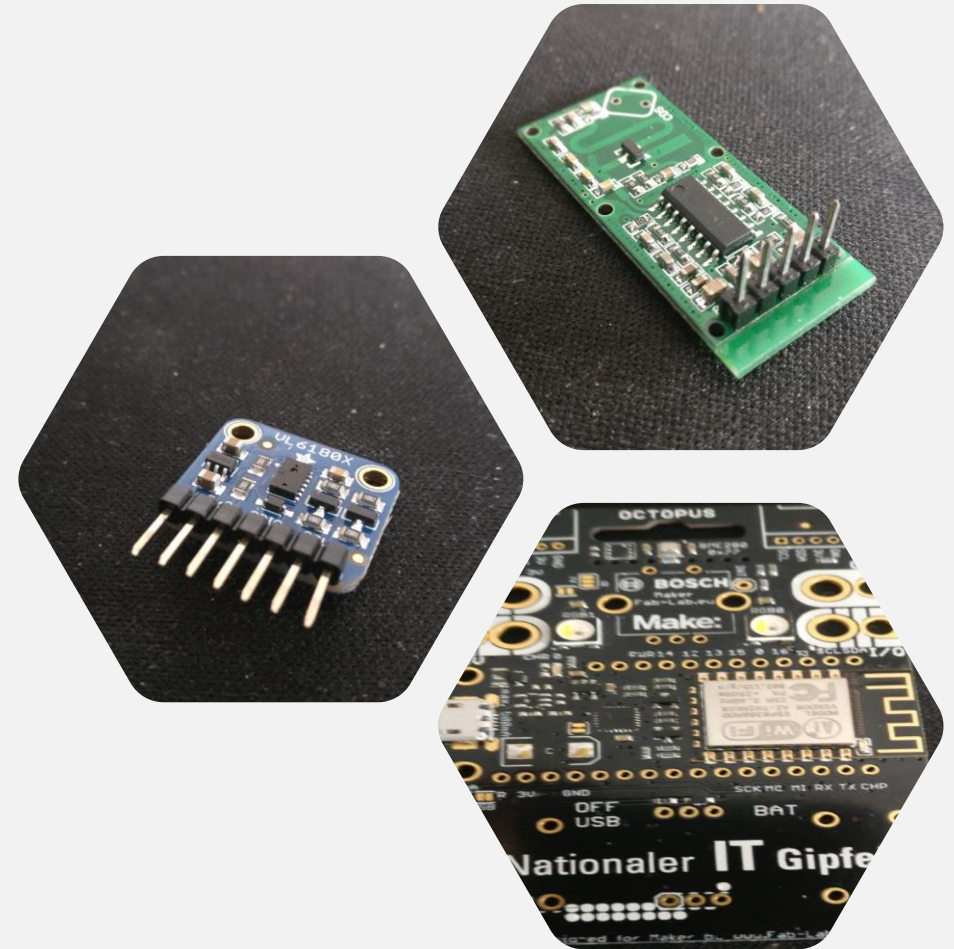
Wieso ein smartes Fenster?

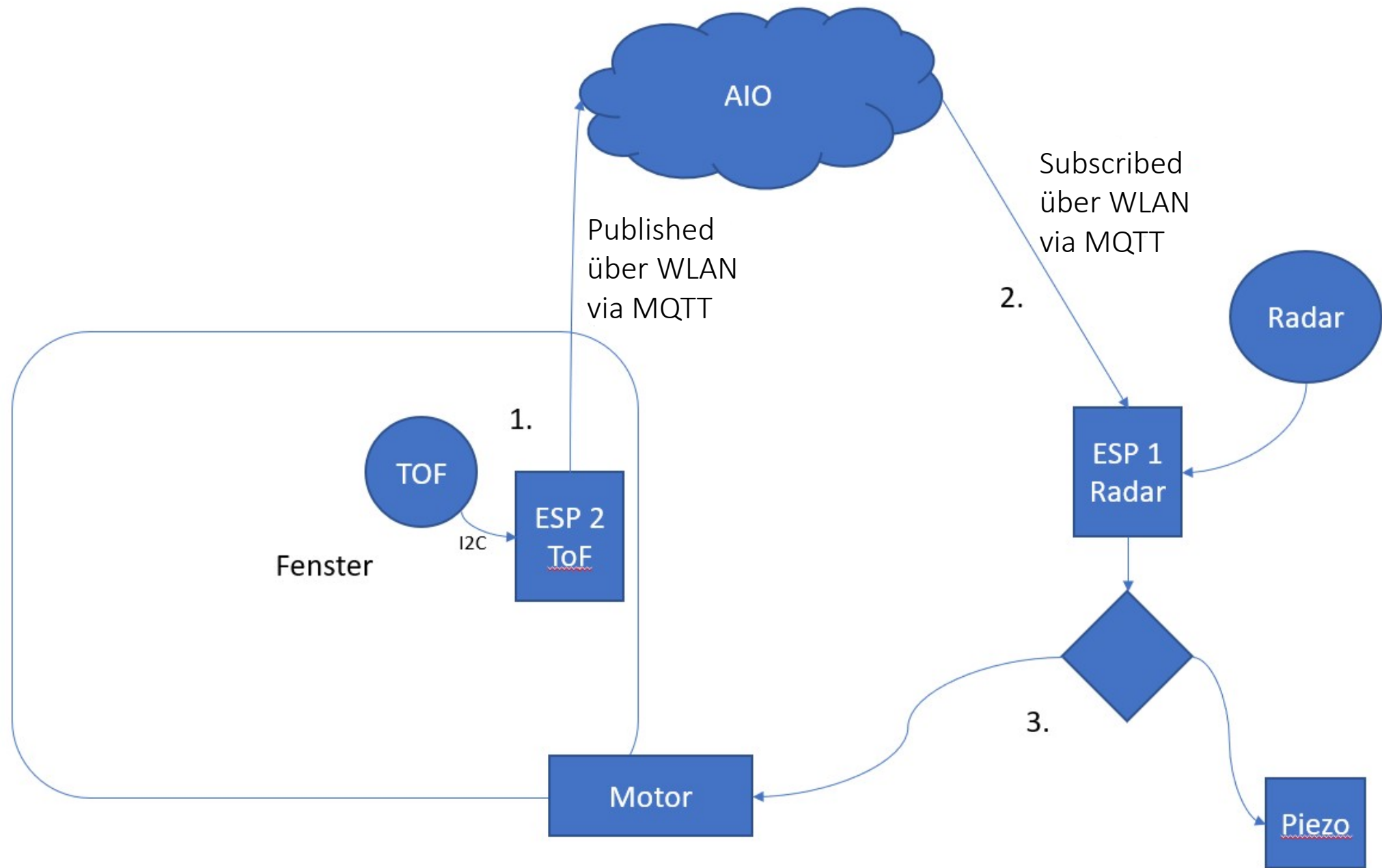


Verwendete Mittel

Was hab ich benutzt um mein Ziel zu erreichen?

- Den Radar Sensor RCWL-0516
- Den Time of Flight Sensor VL6180X
- Sowie 2 ESP 8266 verbaut auf einem IoT Octopus
- Dazu noch Adafruit IO als Online Plattform

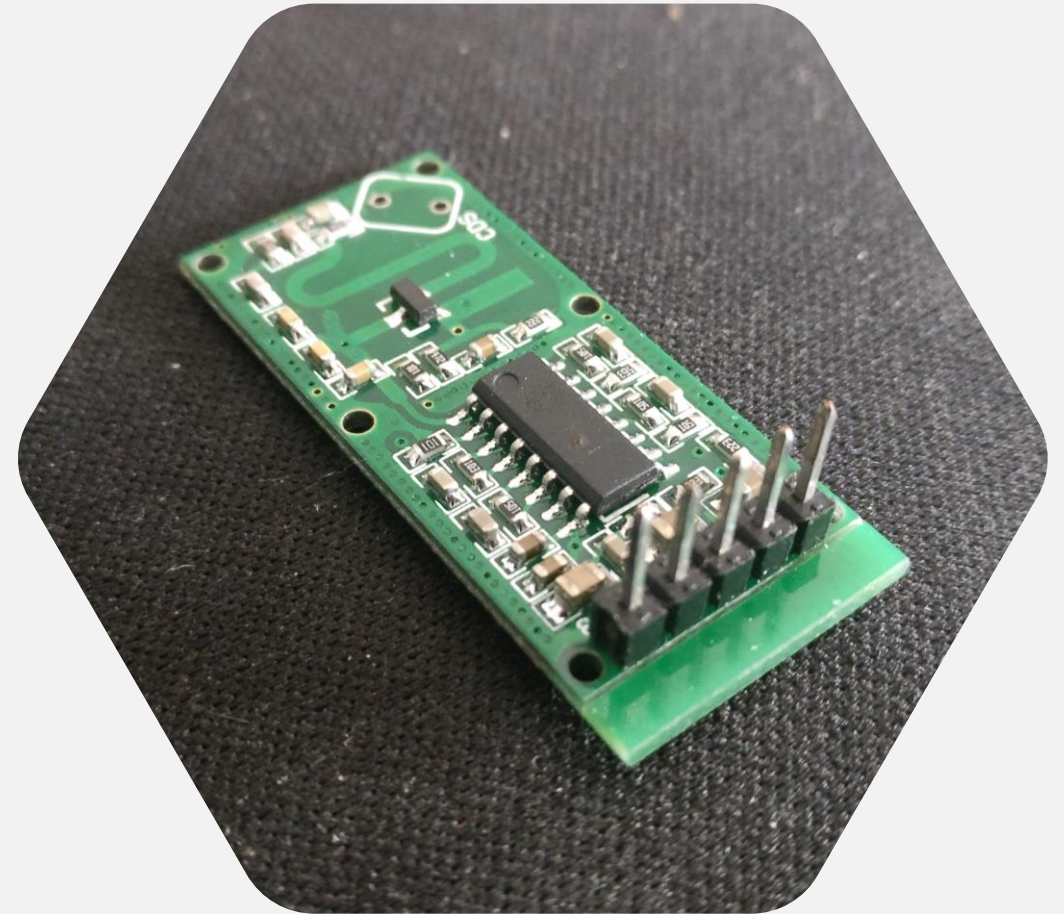
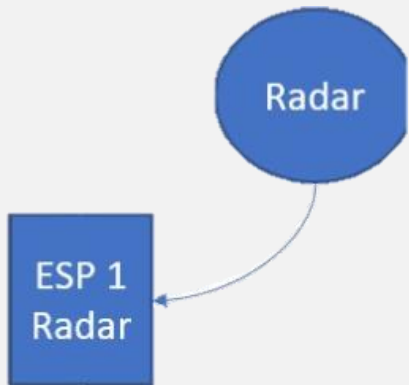




Der RCWL-0516

Oder auch der Radar-Sensor

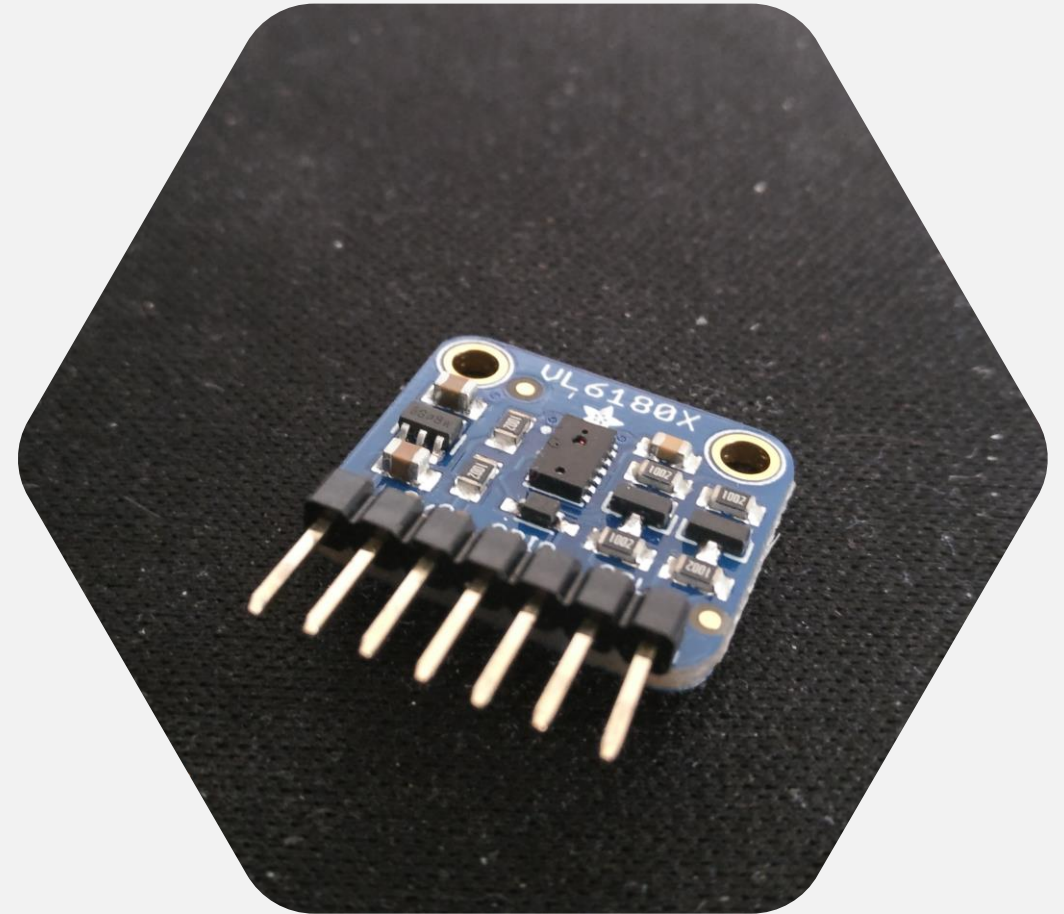
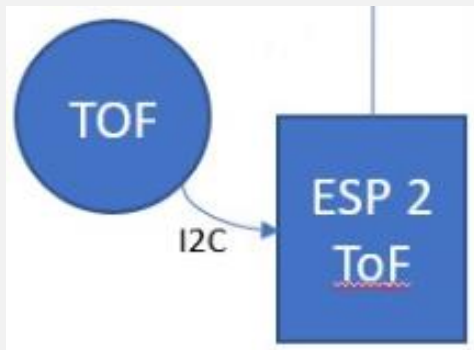
- Relativ neu und noch wenig genutzt
- Hat eine Output Logik von 3,3 Volt
- Benötigt aber 5V Betriebsspannung
- Kann genutzt werden um Bewegung Festzustellen
- Lässt sich nur von wenigen Objekten dabei stören



Der VL6180X

Oder auch Time of Flight Sensor

- Klein und fein
- Kann Distanzen bis zu 200mm messen.
- Nutzt hierfür einen Laser
- Kann zusätzlich auch Helligkeit messen
- 3,3 Volt ausreichend für den Betrieb
- Nutzt I2C zur Übertragung der Sensordaten



Der IoT Octopus

Oder auch ESP8266

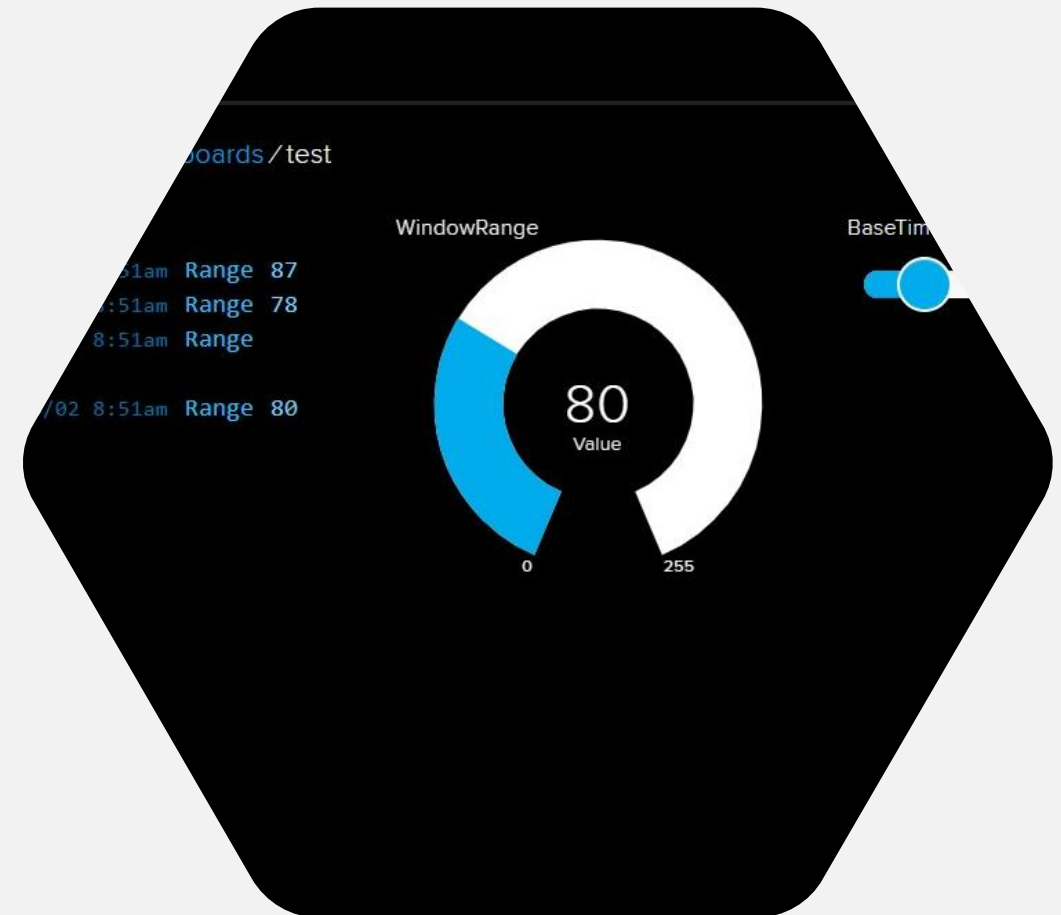
- Der IoT Octopus ist eine Platinen Erweiterung eines ESP8266
- Gedacht für Rapid-Prototyping und das Internet of Things
- Hat dank ESP8266 WLAN an Board.
- Zusätzlich NeoPixel RGB LEDs verbaut
- Direkt über USB anschließ- und programmierbar
- Liefert über eigene Pins nur 3,3V

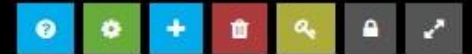


Adafruit IO

Der MQTT Broker

- Webplattform
- Bietet sehr einfach Veranschaulichung von Sensorwerten
- Einfach in das Programm einzubauen.
- Nutzt MQTT ein Urgestein in Maschine to Maschine Communication
- Lässt im Gratis Modus “nur” 30 Uploads pro Minute zu



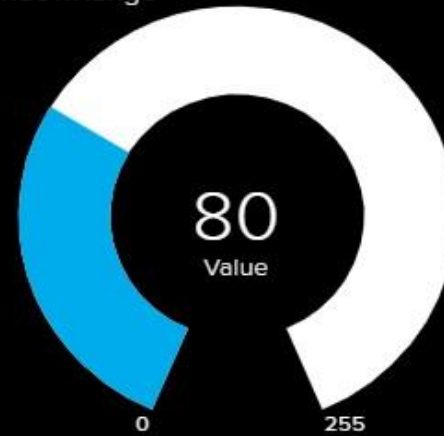


SmartWindow / Dashboards / test

Published Data

2019/04/02 8:51am Range 87
2019/04/02 8:51am Range 78
2019/04/02 8:51am Range
186
2019/04/02 8:51am Range 80

WindowRange

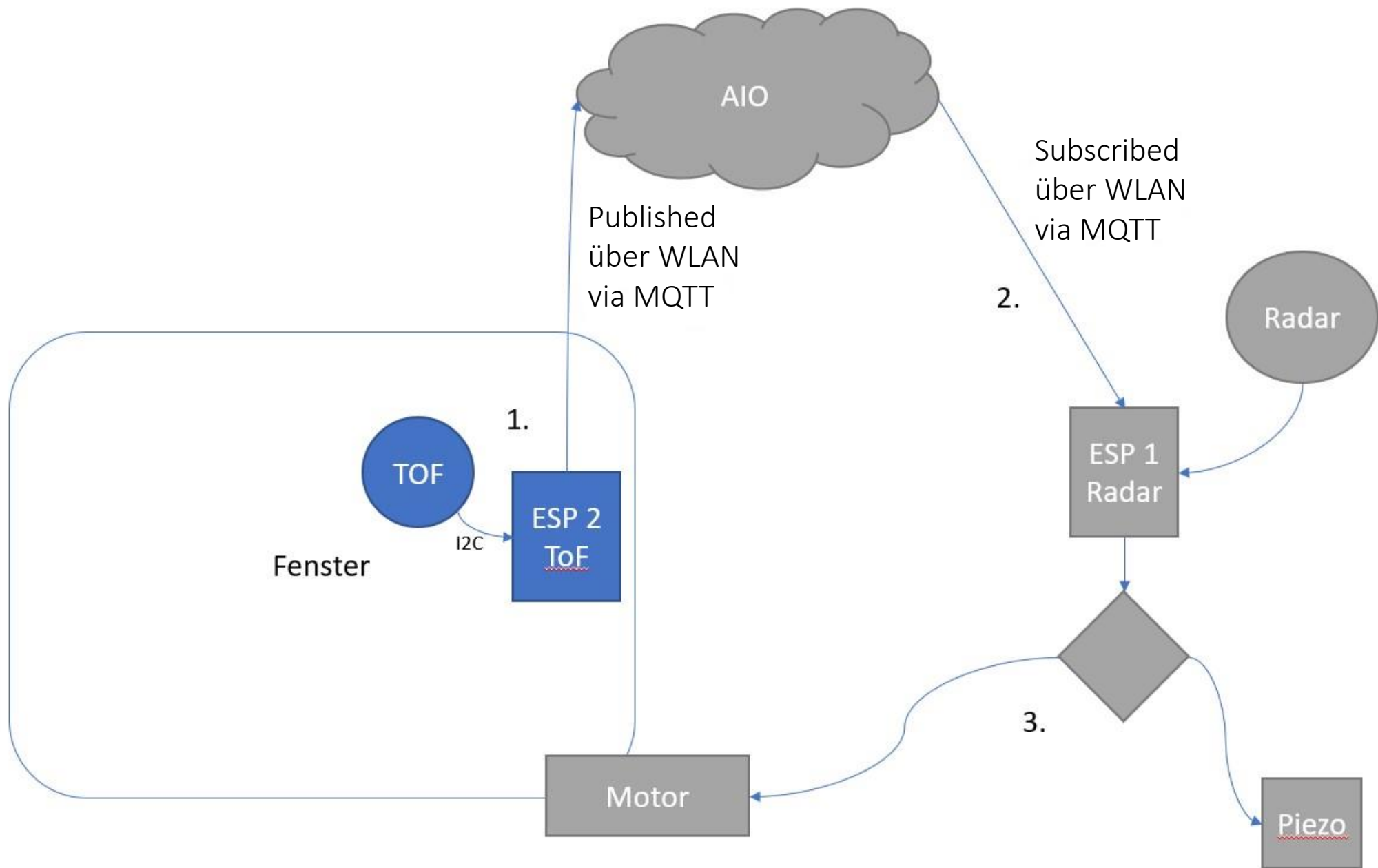


BaseTime



The background features a dark blue field filled with glowing binary digits (0s and 1s) in a light blue color. Several hexagonal shapes of varying sizes are scattered across the image, some containing binary code and others appearing as solid dark blue. A large, dark gray rectangular box is centered in the lower half of the image, containing the title text.

Der Code



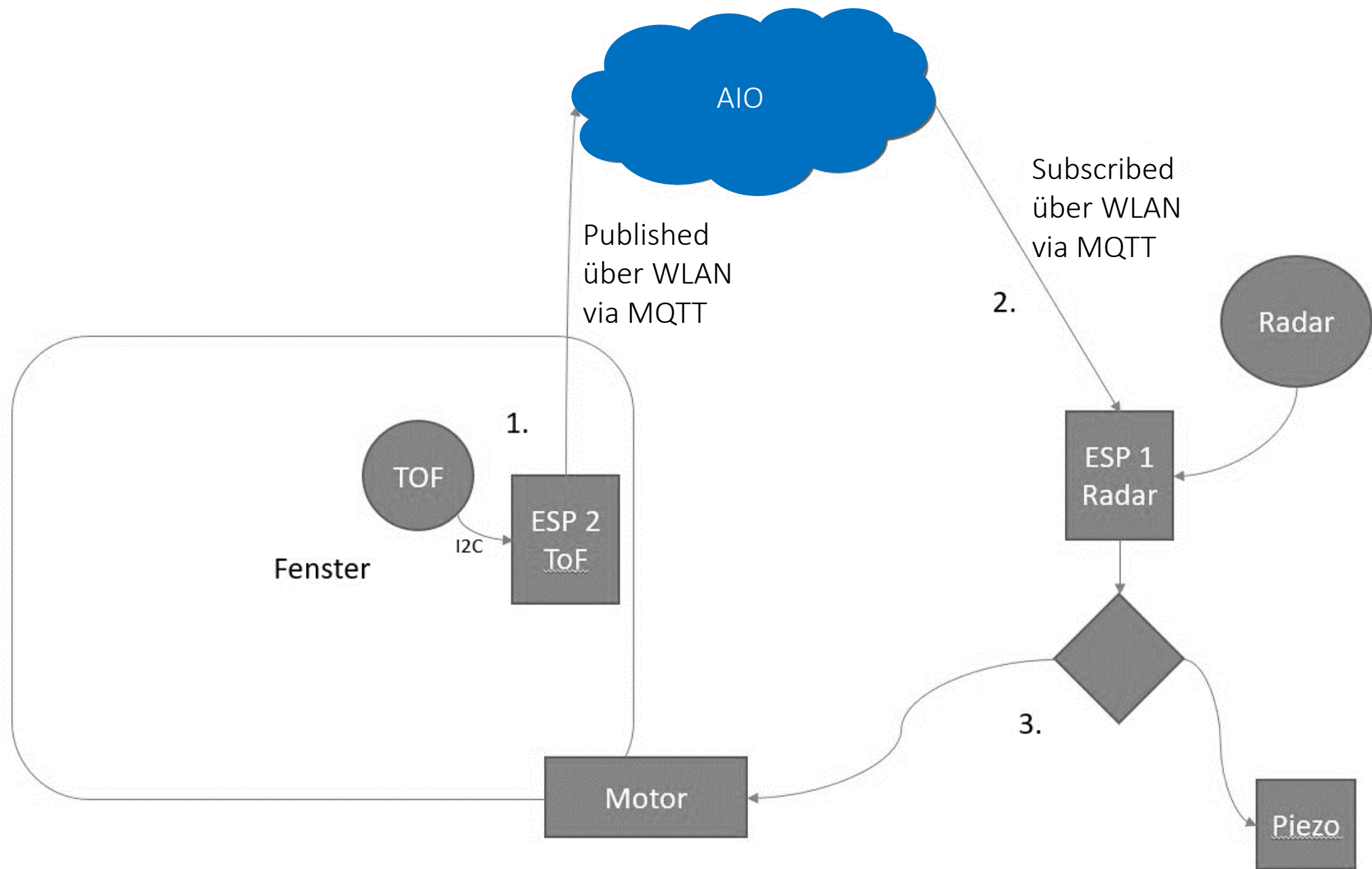
Wie wird der VL6180X angesprochen?

```
// Lib for using I2C
#include <Wire.h>

// Lib for using the time of flight sensor
#include <Adafruit_VL6180X.h>

//TOF object
Adafruit_VL6180X tof = Adafruit_VL6180X();
```

- Adafruit bietet hierfür eine Entsprechende Bibliothek
- Zusätzlich wird noch die Bibliothek `Wire.h` benötigt.
- Erstellen eines Objekts der Klasse `Adafruit_VL6180X`.
- Die in der Klasse definierten Funktionen ermöglichen dann den Zugriff auf die Sensordaten.



Wie wird Adafruit IO eingebunden ?

- Es werden mehrere Bibliotheken benötigt um Adafruit IO in ein Programm einzubinden.
- Des weiteren gibt es eine `config.h`

```
//Lib for using and Connecting to Aidafruit IO
#include <AdafruitIO.h>
#include <AdafruitIO_Dashboard.h>
#include <AdafruitIO_Data.h>
#include <AdafruitIO_Definitions.h>
#include <AdafruitIO_Feed.h>
#include <AdafruitIO_Group.h>
#include <AdafruitIO_MQTT.h>
#include <AdafruitIO_Time.h>
#include <AdafruitIO_WiFi.h>
#include "config.h"
```

Die Konfiguration für Adafruit IO


```
1  #include <Arduino.h>
2  /***** Adafruit IO Config *****/
3
4
5  #define IO_USERNAME  "SmartWindow"
6  #define IO_KEY       "Adafruit IO Key"
7
8
9  #define WIFI_SSID     "test"
10 #define WIFI_PASS     "test1234"
11
12 #include "AdafruitIO_WiFi.h"
13 AdafruitIO_WiFi io(IO_USERNAME, IO_KEY, WIFI_SSID, WIFI_PASS);
14
```

- In `config.h` werden die Zugangsdaten für die Nutzung von Adafruit IO sowie die des WLAN Netzwerkes hinterlegt.
- Dort wird mit diesen Daten ein IO Objekt erstellt das den Zugang zu den Online daten darstellt.

Wie wird Adafruit IO benutzt ?

- Erstellen eines Feed Objekts
 - Repräsentiert als was die Daten Hochgeladen werden
- Start der Verbindung über `io.run()`
- Senden der Daten über die Funktion `save()` des Feed Objekts

```
36 AdafruitIO_Feed *Range = io.feed("range");
37
38 void setup() {
39     last_feed=0;
40     pinMode(shutdownPin, OUTPUT);
41     startPixels();
42     connectAIO();
43     if (!tof.begin())
44     {
45         tofNotFound();
46     }
47 }
```



```
49 void loop() {
50     io.run();
51     if (millis() - last_feed > 7000)
52     {
53         digitalWrite(shutdownPin, HIGH);
54         if (!tof.begin())
55         { ...
56         }
57     }
58     else
59     {
60
61
62         if (tof.readRangeStatus() == VL6180X_ERROR_NONE)
63         { ...
64         }
65
66         if (millis() >= (last_feed + 60000))
67         {
68             print("sending-> ");
69             println(feed);
70             Range->save(feed);
71             last_feed = millis();
72             digitalWrite(shutdownPin, LOW);
73         }
74     }
75 }
```



Handling der Limitierungen von Adafruit IO

- Zeit des letzten Sendens merken
- Um Ungenauigkeit des Sensors auszugleichen wird ein Mittelwert der Daten erstellt
- Sobald eine Minute um ist wird der gemittelte Wert Hochgeladen
- Zeit des letzten Sendens wird auf jetzt gesetzt.

```
f.readRangeStatus()==VL61
```

```
int i=0;  
feed=0;  
while(millis()<=(last_feed+60000))
```

```
{  
    feed+=tof.readRange();  
    i++;
```

```
feed/=i;
```

```
millis()>=(last_feed+60000))
```

```
"sending-> ":println
```

```
define shutdownPin 12
```

```
void tofNotFound();  
void connectAIO();
```

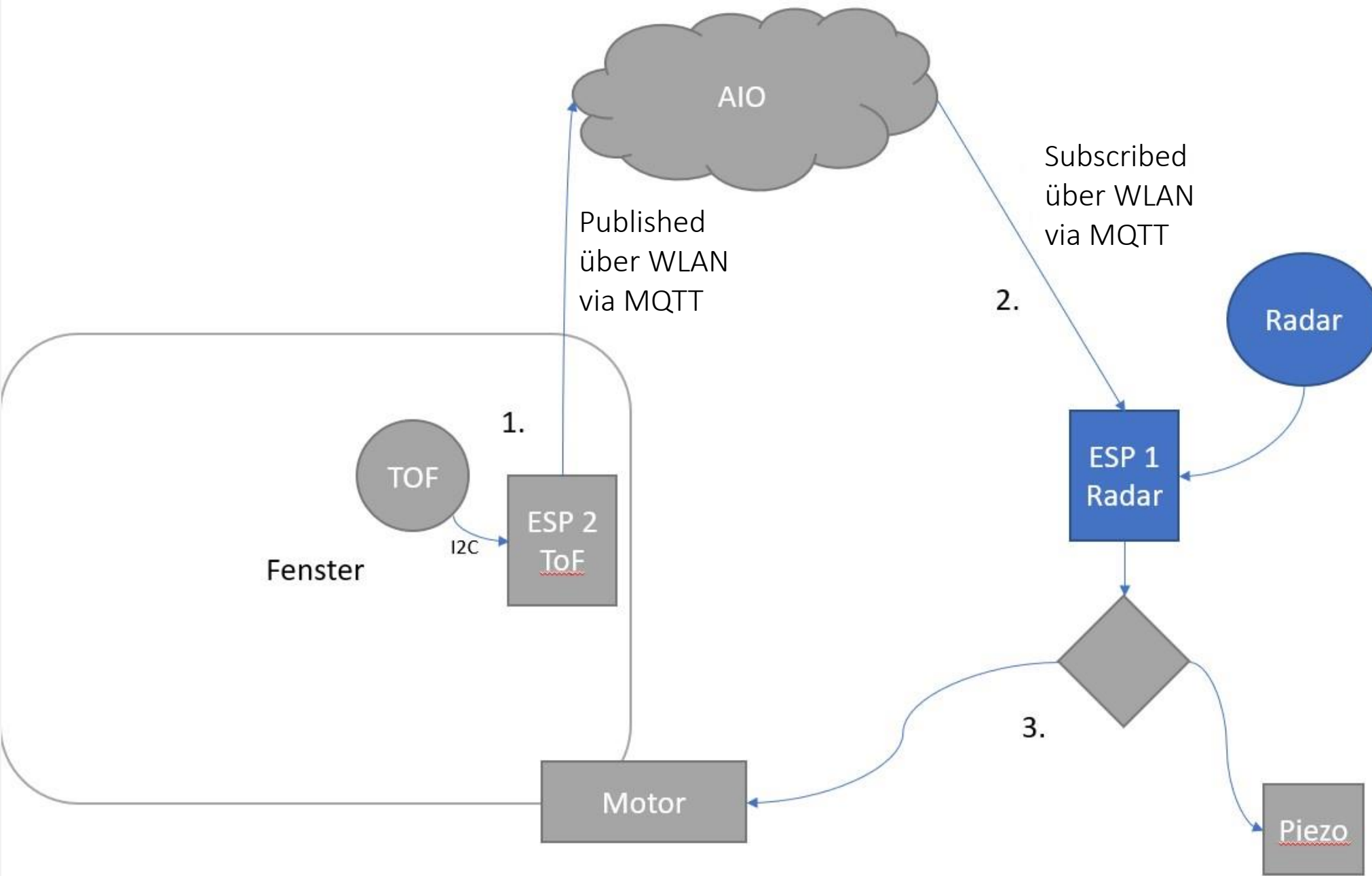
```
unsigned long last_feed;  
unsigned int feed;  
//Object that is used to define  
AdafruitIO_Feed *Range = io.fe
```

```
void setup() {  
    last_feed=0;  
    pinMode(shutdownPin,OU  
    startPixels();
```

```
feed/=i;
```

```
(millis()>=(last_feed+60000))
```

```
{  
    print("sending-> ");println(feed);  
    Range->save(feed);  
    last_feed=millis();  
    digitalWrite(shutdownPin,LOW);
```



Der Code für den Radar Sensor

- Verfügt über keine Bibliothek
- Relativ einfach zu verstehen
3,3V = 1 und 0V = 0.
- Löst einen Interrupt aus der eine Variable verändert

```
void loop() {  
  io.run();  
  range->get();  
  if ((windowRange >= closed+3))  
  {  
    if (millis() >= (lastTime + activeTime))  
    {  
      doSomething(-1);  
    }  
  }  
}  
  
void getTimeInMs(AdafruitIO_Data *data)
```

- Nutzung einer variablen Zeit die über Adafruit IO in einem Slider eingestellt werden kann.
- In dieser Zeit keine Bewegung heißt tue etwas z.B. einen Alarm auslösen.

Thank
You

**Danke für die
Aufmerksamkeit**

Erstellt von Yannick Bayard

Quellen

- <https://giphy.com/gifs/window-mind-vssMehooPRWxi>
- <https://giphy.com/gifs/code-ko7twHhomhk8E>
- <https://media.giphy.com/media/l0MYvqais2XEIWrOU/source.gif>
- <https://giphy.com/gifs/animation-animated-l3q2FnW3yZRJVZH2g>

