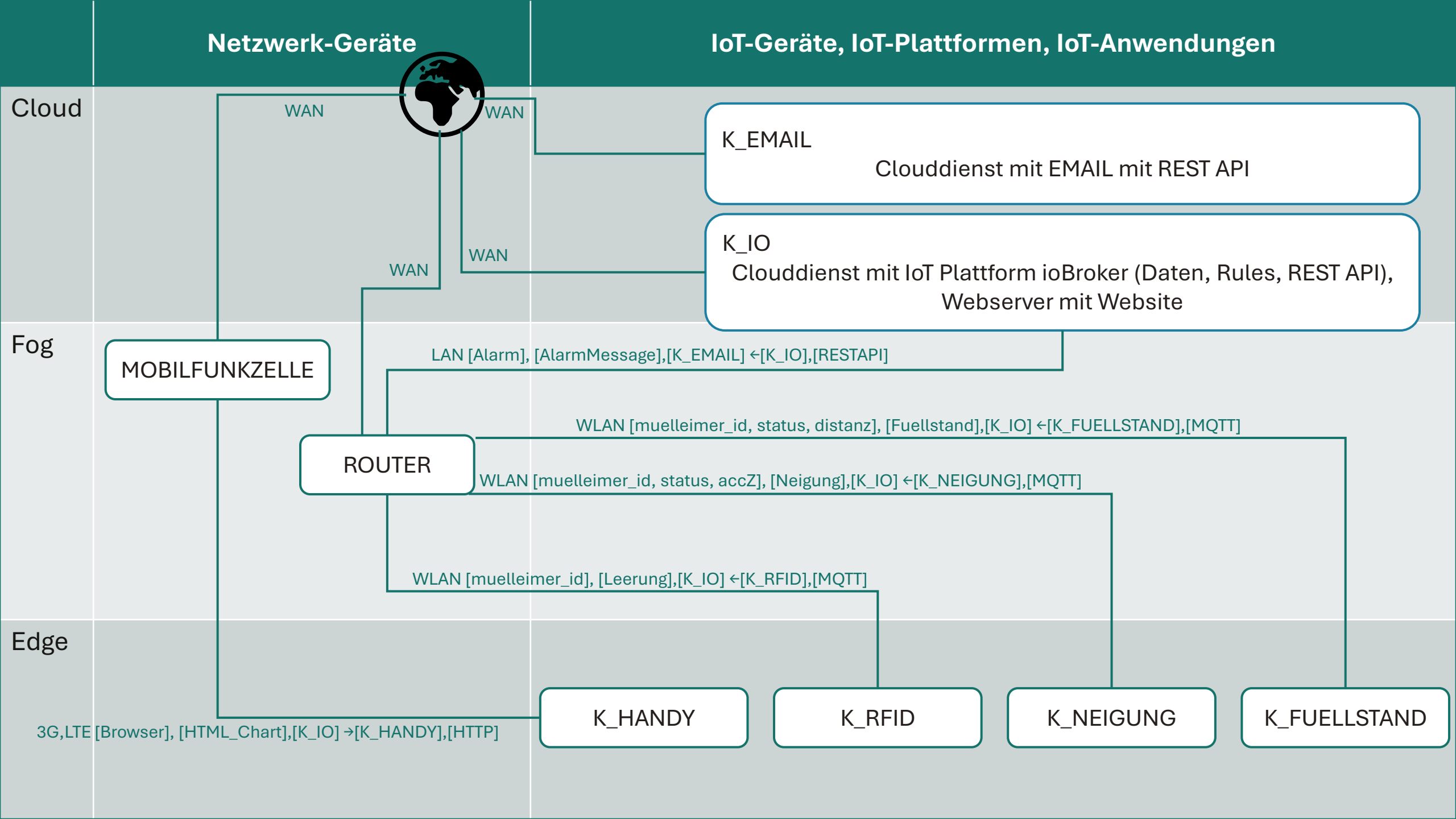
Netzwerk-Geräte
IoT-Geräte, IoT-Plattformen, IoT-Anwendungen
Cloud
Fog
Edge
WAN
WAN
WAN
WAN
K_EMAIL
Clouddienst mit EMAIL mit REST API
K_IO
Clouddienst mit IoT Plattform ioBroker (Daten, Rules, REST API), Webserver mit Website
MOBILFUNKZELLE
ROUTER
LAN [Alarm], [AlarmMessage],[K_EMAIL] ←[K_IO],[RESTAPI]
WLAN [muelleimer_id, status, distanz], [Fuellstand],[K_IO] ←[K_FUELLSTAND],[MQTT]
WLAN [muelleimer_id, status, accZ], [Neigung],[K_IO] ←[K_NEIGUNG],[MQTT]
WLAN [muelleimer_id], [Leerung],[K_IO] ←[K_RFID],[MQTT]
3G,LTE [Browser], [HTML_Chart],[K_IO] →[K_HANDY],[HTTP]
K_HANDY
K_RFID
K_NEIGUNG
K_FUELLSTAND

Mülleimer-ID Scannung

Hardware

- Arduino, RFID, Akku

Daten

- [Leerung]: Sensor an Arduino

M2M-Kommunikation

- [MQTT]: MQTT-Publisher (MQTT-Broker von [K_IO])

Applikationen/Dienste/Bibliotheken

```
// Ermöglicht die WLAN-Funktionalität
  #include <ESP8266WiFi.h>

// MQTT-Client-Bibliothek
  #include <PubSubClient.h>

// RFID-Reader MFRC522 Bibliothek
  #include <MFRC522.h>

// Serielle Peripherie-Schnittstelle für RFID
  #include <SPI.h>
```

Events + technische Programmabläufe

- [muelleimer_id]: Wenn der RFID-Chip einer Mülltonne gescannt wird, sende diesen per [MQTT] an ioBroker [K_IO]

NEIGUNGS-MESSER [K_NEIGUNG]

Hardware

- Arduino, MPU6050, Akku

Daten

- [Neigung]: Sensor an Arduino

M2M-Kommunikation

- [MQTT]: MQTT-Publisher (MQTT-Broker von [K_IO])

Applikationen/Dienste/Bibliotheken

```
// Für WLAN-Funktionalität
  #include <ESP8266WiFi.h>

// Für MQTT-Kommunikation
  #include <PubSubClient.h>

// Für I2C-Verbindung zum MPU6050
  #include <Wire.h>

// Adafruit-Bibliothek für den MPU6050
  #include <Adafruit_MPU6050.h>

// Einheitliche Sensor-API von Adafruit
  #include <Adafruit_Sensor.h>
```

Events + technische Programmabläufe

- [muelleimer_id, status, accZ]: Wenn der Status („umgekippt“, „steht“) einer Mülltonne sich ändert, sende diesen per [MQTT] an ioBroker [K_IO]

FUELLSTANDS-MESSER [K_Fuellstand]

Hardware

- Arduino, HC-SR04, Akku

Daten

- [Fuellstand]: Sensor an Arduino

M2M-Kommunikation

- [MQTT]: MQTT-Publisher (MQTT-Broker von [K_IO])

Applikationen/Dienste/Bibliotheken

```
// Für WLAN-Funktionalität
  #include <ESP8266WiFi.h>

// Für MQTT-Kommunikation
  #include <PubSubClient.h>
```

Events + technische Programmabläufe

- [muelleimer_id, status, distanz]: Wenn der Status („nicht messbar“, „voll“, „halbvoll“, „leer“) einer Mülltonne sich ändert, sende diesen per [MQTT] an ioBroker [K_IO]

FOG-ZENTRALKE [K_IO]

Hardware

- Arduino

Daten

- [status]: ioBroker speichert Status der Neigung (Echtzeit)
- [HTML_Chart]: Webserver/PHP mit HTML-Website für [K_Handy]
- [AlarmMessage]: Text für E-Mail-Nachricht bei Alarm

M2M-Kommunikation

- [MQTT]: MQTT-Broker und –Client
- [RESTAPI]: HTTP-Aufruf im RESTAPI-Format (z.B. [K_EMAIL])
- [HTTP]: HTTP-Aufruf

Applikationen/Dienste/Bibliotheken

- IoT-Plattform ioBroker, Webserver Lightpd mit PHP

Events + technische Programmabläufe

- [AlarmCheck]: ioBroker Rule (Javascript) überprüft bei jedem neuen Wert, ob der Wert „umgekippt“ entspricht. Wenn ja, dann wird eine E-Mail Benachrichtigung per RESTAPI ausgeführt
- [Browser]: Aufruf der Webseite mit E-Mail Nachricht [K_Handy]. PHP ruft per [RESTAPI] aktuelle Daten und Vergangenheitsdaten bei ioBroker ab und verknüpft diese mit einem HTML-Template zu einem Chart. Der Webserver liefert das finale HTML per [HTTP] aus.