DAS – Distributed Alarm System

# Background

An alarm system for home use consists of several sensors that are linked to a central unit. The coupling can be hardwired or wireless. Typically, in addition to sensors, there is also a device used to activate and deactivate the alarm.

The central alarm unit stores the status of all sensors and contains logic that determines whether the status of the system should result in an alarm state. Alarms are used to start one or more acts. If the alarm system lacks external connection, the typical act is to start a siren.   
If the central unit has external connections, alarms can be used to trigger messages to an emergency center and/or a mobile phone.

The project Distributed Alarm System (DAS) aims to explore the possibility of building an alarm system without any local central unit. Each individual sensor is instead connected wirelessly directly to a cloud service. This cloud service contains logic to assess whether system status should result in an alarm and what actions should be carried out when alarms are triggered.

There are several potential benefits to distributed, cloud-based alarm systems:

* It is not possible to incapacitate the whole system by sabotaging the central unit.
* It is easy to create communication redundancy by allowing the sensors to communicate through different communication routes (WiFi, Radio link, etc)
* The trend is that hard wired sensors are replaced by wireless sensors in home alarm systems. The driving force behind this is that the installation of wired sensors is more expensive than the sensors themselves. As a result, the cost of sensors will be the same in a cloud-based alarm system as in a classical home alarm system. The need for a costly central unit is though eliminated in a cloud-based alarm system.
* Central units are limited in terms of how large alarm systems can be. This is due to limitations in memory and processor capacity. This limitation does not exist in cloud-based systems.

The obvious downside of a cloud-based alarm system is that if the cloud service goes down, the alarm is not functional.

# Intention and Limitations

The purpose of the project is not to implement a full alarm system. Instead, the aim is to investigate and demonstrate the mechanisms needed in an alarm system built on IoT components. Several restrictions are therefore accepted:

A commercial alarm system must fulfill a very high level of security both for authentication, authorization and data transfer. This is difficult to achieve with ubiquitous IoT devices. These currently lack security cores and cannot handle asymmetric encryption satisfactorily. In order to be able to use commonly available IoT-devices, the project does not need to meet any specific security level.

A commercially distributed system must be easy to initiate and configure. This leads to the need for good configuration tools. For reasons of time, the project refrains from implementing this aspect.

To achieve the full advantage of a distributed alarm system, several different types of communication should be available between IoT devices and the cloud, as well as between the cloud and alarm receivers (alarm centers, individual alarm owners). For time reasons, the project is not required to examine multiple method of communication.

# Scope

The following requirements shall be met by the project:

The system shall contain at least the following IoT devices:

* A unit to arm and secure the alarm system.
* A motion detector.
* A distance detector for securing e.g. entrances
* A low-sound siren for alarm indication.

The system shall contain an external status component where sensor status is reported. This should be implemented as a cloud service.

The system shall contain an external logic component that interprets status and triggers alarms. This can be implemented in any way.

The system shall contain a mechanism that allows the logic component to communicate with an external operator. Examples of acceptable communication methods are as simple tools as email, SMS, etc.

The system may contain a monitoring component that can be used to manually monitor sensor statuses and alarm status. This component can be implemented as part of the logic component or as a standalone part.

Alarm Logic

Component

External Communicator Component

System Status Broker

Status Monitor (optional)

Alarm Siren Unit

Distance Sensor  
Unit

Motion Detection Unit

Arming Unit

# Proposed Technology Platform

As the aim of the project is to investigate which components can be used for a distributed alarm system. A few proposals on platforms are listed solely as a starting point. It says the project is free to deviate from these proposals if the time or availability so require. It also says the project is free to deviate from the proposals for more optimal platforms found.

The status component is implemented on AWS or Azure (if possible). Alternatively, a cloud based MQTT server may be used as a status broker.  
If it proves impossible to use cloud-based platforms due to time or cost reasons, a local server can be used to simulate a cloud platform.

The IoT devices are implemented using ESP2866 units.

Arduino is used as a development system for the IoT devices.

# Topic Definition

Regardless of which system status server that is selected, the technical communication base is likely to be MQTT. In MQTT clients communicate with a sever (broker). The clients publish information classified by so called topics. These may be seen as labels for the information messages that are published. The clients may also subscribe to topics. Whenever a topic is updated on the server, all subscribers are notified, and the new message is relayed to them.

|  |  |  |
| --- | --- | --- |
|  | **Publish** | **Subscribe** |
| **Device** | **Topic, Type, Value** | **Topic, Type, Value** |
| Arming Unit | alarm/button, bool, 0 or 1 | alarm/active  alarm/entry\_alarm |
| Motion Detection Unit | alarm/pir, bool, 0 or 1 | alarm/active  alarm/entry\_alarm |
| Distance Sensor Unit | alarm/pir, bool, 0 or 1 | alarm/active  alarm/entry\_alarm |
| Alarm Siren Unit |  | alarm/active  alarm/entry\_alarm |
|  |  |  |
| **Component** |  |  |
| Alarm Logic Component | alarm/active, bool, 0 or 1 alarm/entry\_alarm, bool, 0 or 1 | alarm/active  alarm/entry\_alarm alarm/pir  alarm/siren  alarm/button |
| External Communicator Component |  | alarm/entry\_alarm |

# Hardware Units

## Arming Unit

A circuit board

Description automatically generated

Base: Wemos D1 Mini controller board

Arming actuator: Wemos Shield, 1-ButtonPush Button, PIN D3

Status Indicator: Custom made Wemos Shield with a red and a green LED and battery power supply connection.

Wemos Dual module Board

## Motion Detection Unit

A circuit board

Description automatically generated

Base: Wemos D1 Mini controller board

Motion detector: Wemos PIR module shield

Status Indicator: Custom made Wemos Shield with a red and a green LED and battery power supply connection.

## Distance Sensor Unit

A close up of electronics

Description automatically generated

Base: Wemos D1 Mini controller board

Distance Sensor: Custom Wemos shield with HC-SR04 ultrasonic distance sensor

Status Indicator: Custom made Wemos Shield with a red and a green LED and battery power supply connection.

## Alarm Siren Unit

A circuit board

Description automatically generated

Base: Wemos D1 Mini controller board

Buzzer: Wemos D1 mini Buzzer Shield, PIN D5

Status Indicator: Custom made Wemos Shield with a red and a green LED and battery power supply connection.

## Hardware pinout

A circuit board

Description automatically generated

D0 = Alarm LED Out (red)  
D1 = LCD SCL  
D2 = LCD SDA  
D3 = SR04-Trigger Out  
D4 = Status LED Out (green)  
D5 = Type bit 1 In  
D6 = PIR In  
 Button In   
 SR04 Echo In  
D7 = Type bit 0 In  
D8 = Siren Out

Module types as decoded by type bits (D5 D7):

00 = Button  
01 = HC-SR04 (Ultrasonic Sensor)  
10 = PIR motion detection module  
11 = Siren module

# Alarm Logic Component

This part of the system is implemented as a python program.

The program can run as a cloud process or on any internet connected device, such as a PC or in a cloud service.

The logic is simple:

The program subscribes to all alarm signals.

When the program detects that the alarm actuator has been pressed it toggles the alarm/activated topic value in the system MQTT broker.

When the program detects that an alarm/pir topic is posted with the value 1 (ON), it checks if the alarm is activated. If this is the case it sets the alarm/entry\_alarm topic in the MQTT server to 1 (ON).

# Status Monitor

This part of the system is implemented using an MQTT Android app. The alarm/entry\_alarm state of the system as well as the alarm/armed state are displayed.

As a bonus, it is possible to toggle both system states in the Android app.A screenshot of a cell phone

Description automatically generated