FH CAMPUS WIEN
UNIVERSITY OF APPLIED SCIENCES

Design document

Room Ventilation

11.10.2024

Group Nr. 3

Timur Ali Basnakajev
Tobias König
Jessica Marban
Patrick Neumann

Contents

1 Project description

1.1 Problem description

The idea of our project is to build an automatic room ventilation. This could be used in a classroom or an office, for example. When the CO2 level in a room reaches a certain level, a window should automatically open and a fresh air fan should be switch on. After a certain lower CO2 level is reached and a certain run-on time has elapsed, the window should be closed and the fan switched off again.

The system is divided into 2 larger areas, one of which is a Raspberry Pi and the other is an Arduino. These two components are supposed to communicate with each other and exchange data via a secure Bluetooth connection.

1.2 Arduino

The project part of the Arduino consists of the following hardware:

1. Arduino Nano 33 IoT
2. MiCS-VZ-89TE CO2 Sensor
3. cabling + Error-LED

The main task of the Arduino is data acquisition and data processing. It should measure the data via the co2 sensor, check its validity and then interpret it. Based on the values, their history and certain thresholds, the Arduino sends its current status to the Raspberry Pi every 10 seconds. This can be IDLE, OPEN or CLOSE. The status OPEN and CLOSE are sent once when the window is to be opened/closed and the fan is to be switched on/off.

1.3 Raspberry Pi

The project part of the Raspberry Pi consists of the following hardware:

1. Raspberry Pi 4 Model B 8GB
2. 2x Mini Stepper Motor Kit ULN2003A with 28BYJ-48
3. potentiometer
4. cabling + Error-LED

On the other hand, the Raspberry Pi is mainly responsible for device control. It receives the status of the Arduino every 10 seconds and works based on it. When the status is OPEN or CLOSE, it controls the motor for the window and the motor for the fan and turns them on/off. The potentiometer is measuring the current position of the window in order for the Raspberry Pi to decide weather the window was opened/closed successfully.

1.4 Schema

In the following diagram you can see the planned structure.

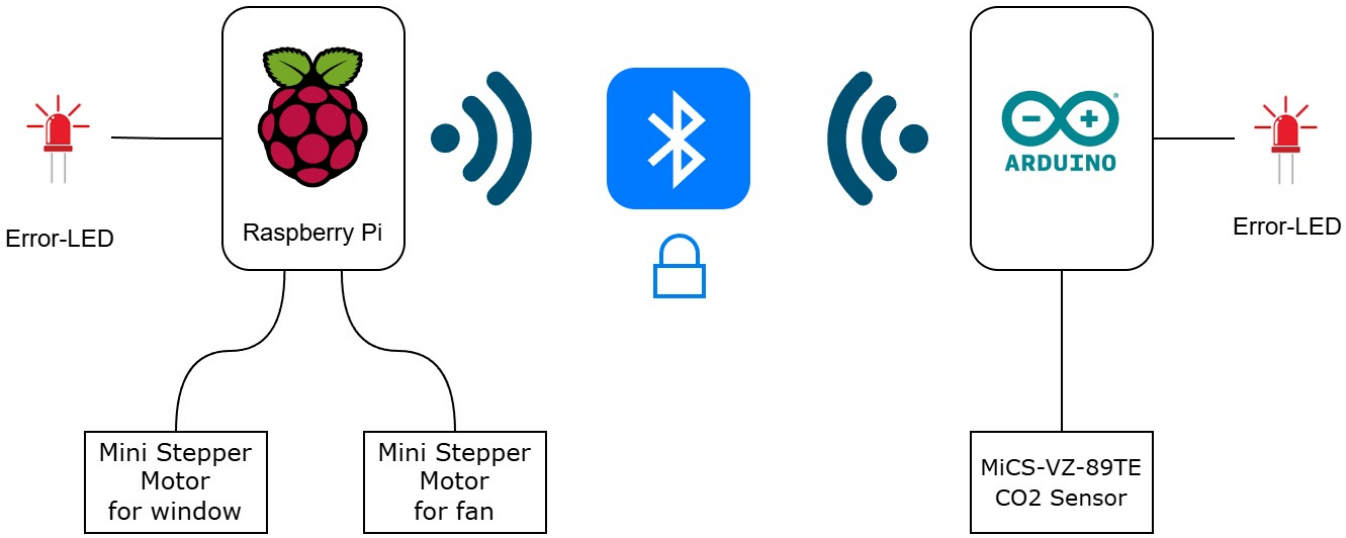

Figure 1: system description

2 Software architecture and design

2.1 Software modules

Safety related modules

1. **Window Control Module (Raspberry Pi):**
   Description: This module receives the status (e.g., OPEN, CLOSE, IDLE) from the Communication Module and controls the electric motor to open or close the window based on the received status. It ensures safe operation by detecting obstacles or problems during movement.
   Functions:

   - Receive status (e.g., OPEN, CLOSE, IDLE) from the Communication Module.
   - Open/close the window using the stepper motor based on the status.
   - Monitor motor movement and detect deviations during window operation.
   - Stop motor and trigger an error state if a problem is detected.
   - Report errors through LED indicators.

   Data: Stepper motor position, window state, status.
   Requirements: 4. (Operating the window), 1.4. (Connection retries and email alerts), 3.1. (Acknowledgment of messages).

2. **Value Interpretation Module (Arduino):**
   Description: This module validates the CO2 sensor readings and determines whether the values are within a valid range. Furthermore it compares the current value with the average of the last measured values and sets the status based on that. The status (OPEN, CLOSE, IDLE) is then sent to the Raspberry Pi via the Communication Module.
   Functions:

   - Measure CO2 concentration using the MH-Z19 sensor.
   - Validate CO2 readings by checking for deviations from the average of previous readings.
   - Determine status (OPEN, CLOSE, IDLE) based on validated values.
   - Send status to the Raspberry Pi via the Communication Module.

   Data: Raw sensor data, validated status (OPEN, CLOSE, IDLE).
   Requirements: 2. (Invalid Value Handling), 7. (Status communication).

Security related modules

1. **Communication Module (Raspberry Pi and Arduino):**
   Description: This module handles the communication between the Raspberry Pi and the Arduino, ensuring secure and reliable message transmission, including handling connection retries and managing encryption keys.
   Functions:

   - Establish Bluetooth communication between Raspberry Pi and Arduino.
   - Implement acknowledgment of messages between devices.
   - Handle connection retries and trigger alerts after failed retries.
   - Implement Diffie-Hellman key exchange for secure communication setup.
   - Perform encryption and decryption of messages using AES-CBC-128.
   - Using HMAC for Data-Integrity and Authentication

   Data: Encrypted communication data, status messages, acknowledgment messages.
   Requirements: 3. (Communication between devices), 1. (Spoofing and key exchange), 4. (Information Disclosure), 1.4. (Email alerts on failed retries).

2. **System Hardening Module (Raspberry Pi):**
   Description: This module focuses on securing the Raspberry Pi against cyber threats by implementing firewall rules and monitoring suspicious activity.
   Functions:

   - Manage user authentication and access control.
   - Monitor and log suspicious activity.

   Data: Security logs, access control records.
   Requirements: 6. (Tampering protection), 3..3 (Repudiation logging).

Modules with no influence on Safety and Security

1. **Fan Control Module (Raspberry Pi):**
   Description: This module controls the fan based on the status received from the Communication Module. It turns on the fan when instructed and turns it off when instructed.
   Functions:

   - Turn on the fan when receiving an OPEN status.
   - Turn off the fan when receiving a CLOSE status.

   Data: Status, fan state.
   Requirements: 8. (Fan activation on OPEN status), 9. (Fan deactivation on CLOSE status).

2. **System Monitoring Module (Raspberry Pi):**
   Description: This module handles general system diagnostics like checking CPU temperature and memory usage. It ensures that non-critical parameters are monitored for optimal system performance.
   Functions:

   - Log and monitor system health, such as CPU temperature and memory usage.
   - Log non-critical warnings for maintenance purposes.

   Data: System diagnostics logs, performance metrics.
   Requirements: 1. (The system should be installed in an inaccessible location to ensure reliability).

2.2 Libraries

The following libraries are used to interface with hardware components and to implement the functionality described above:

- **MiCS-VZ-89TE Library (Arduino):** 'MICS-VZ-89TE' or 'SoftwareSerial' - to read CO2 data via UART from the MH-Z19 sensor and convert it into ppm values.
- **Stepper Motor Control Library (Raspberry Pi):** 'wiringPi' or equivalent C libraries for controlling the stepper motor for window movement.
- **Communication Libraries:**
  - Raspberry Pi: 'bluez' (C library) - for managing Bluetooth communication.
  - Arduino: 'Arduino BluetoothSerial' - for establishing and maintaining Bluetooth connections.
- **Encryption Libraries:**
  - Arduino & Raspberry Pi: 'AESLib' for or encrypting and decrypting communication data

2.3 Interrupts

Definition of priorities:

- **Priority 1:** Communication failure detection interrupt between the Raspberry Pi and Arduino to ensure immediate action if data transfer fails.
- **Priority 2:** Window control interrupts for problem detection during motor operation to prevent damage.
- **Priority 3:** Status validation on the Arduino to ensure accurate CO2 evaluation.

2.4 Pinout

- **CO2 Sensor (MiCS-VZ-89TE - Arduino):** Connected via UART for analog data reading.
- **Window Motor Control (Raspberry Pi):** Connected to GPIO pins for controlling the motor.
- **Fan Control (Raspberry Pi):** Connected to GPIO pins for fan activation.
- **Error Indicators (Raspberry Pi & Arduino):** LEDs connected to GPIO pins for status indication.

3 Program flowchart

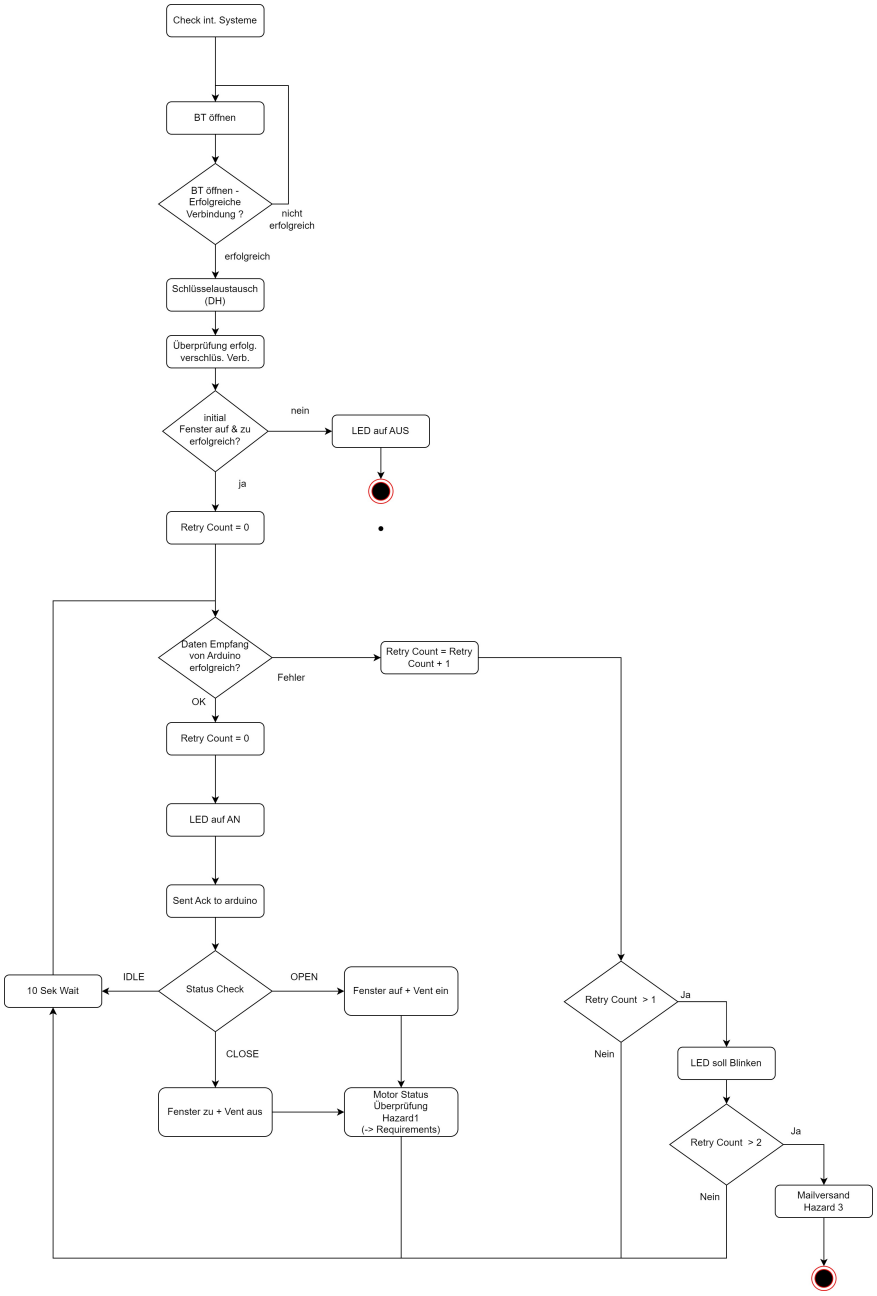

Figure 2: Rasperry PI

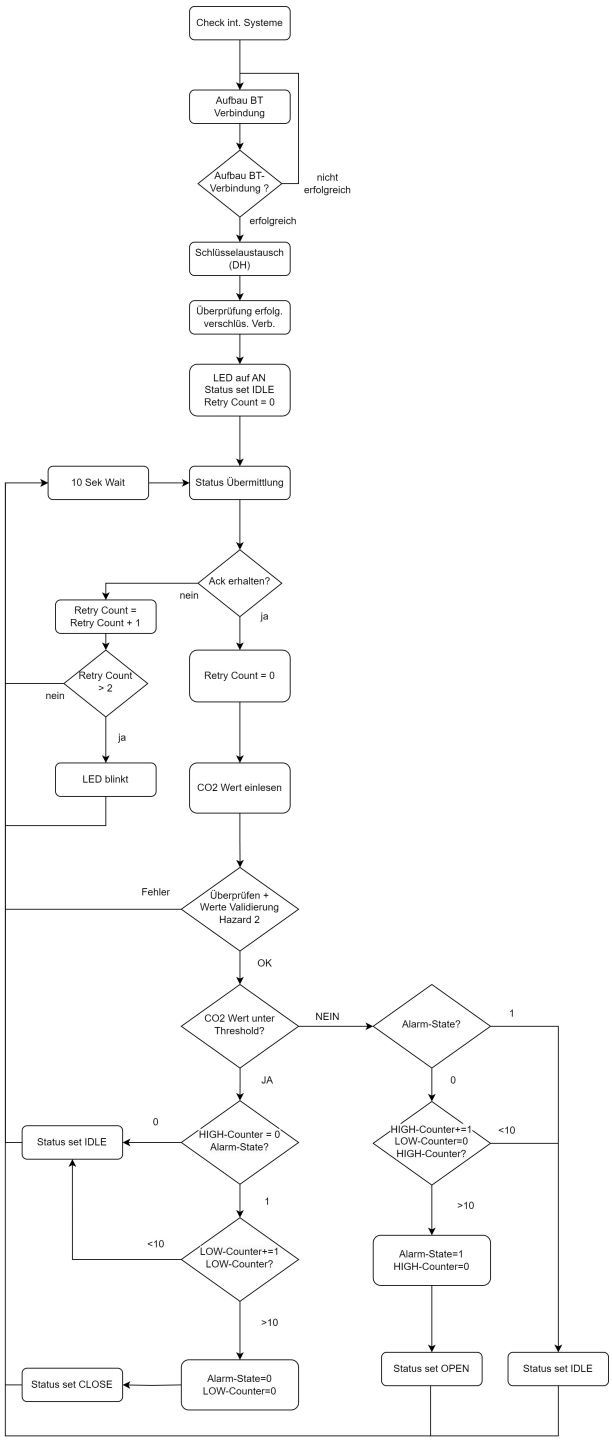

Figure 3: Arduino

4 Hazard identification

4.1 Identified hazards and countermeasures

- **Hazard 1:**
  Co2 sensor delivers incorrect values 2.

  Countermeasures 1:
  All measured values are compared with the average of the previously measured values. If a certain threshold for this difference is exceeded, the value is declared invalid.

- **Hazard 2:**
  The window is stuck and does not open or close 4.

  Countermeasures 2:
  The position of the stepper motor is measured after the opening or closing process has been completed and compared with the default value of a closed or opened window.

- **Hazard 3:**
  Arduino fails 1.

  Countermeasures 3:
  The Raspberry Pi is sending an email to the system administrator after connection-retries with the Arduino are unsuccessful for 30 seconds.

4.2 Identified hazards without countermeasures

- Hazard 1:

  The Co2 sensor fails and does not provide data anymore.

- Hazard 2:

  The power supply fails on the Arduino or the Raspberry Pi.

- Hazard 3:

  The engine for the window fails.

5 Threat identification

5.1 Identified threats and countermeasures

1. **Threat 1:** An attacker can make a log lose or confuse data.

   Mitigation: There must be logs with timestamps.

2. **Threat 2**: An attacker can alter and manipulate the data that is sent back and forth. (Mitm-attack)

   Mitigation: Each message between the Arduino and Raspberry Pi must be accompanied by an HMAC.

3. **Threat 3**: An Attacker carry out a DoS attack on the Raspberry Pi.

   Mitigation: The Raspberry Pi must accept no more than 1 message every 9 seconds from the Arduino to prevent flooding attacks.

5.2 Identified threats without countermeasures

1. **Threat 1:** An attacker can carry out a DoS-attack on the LAN connection of the raspberry pi in order to prevent it from sending an email.

2. **Threat 2:** An attacker can provide or control state information.

3. **Threat 3:** An attacker can replay data without detection.

6 Requirements

6.1 Safety related requirements

1. Requirement: Connection Handling

   Requirement:
   If any error occurs during program start at the Arduino, this must be indicated by the Error-LED (LED OFF)

   1.1. Requirement:
   If no error occurs during program start at the Arduino, this must be indicated by the Error-LED (LED ON)

   1.2. Requirement:
   If no error occurs during program start at the Raspberry Pi, this must be indicated by the Error-LED (LED ON)

   1.3. Requirement:
   If any error occurs during program start at the Raspberry Pi, this must be indicated by the Error-LED (LED OFF)

   1.4. Requirement:
   If the Arduino stops communicating with Raspberry Pi, the Raspberry Pi must send an email to the system administrator after connection-retries are unsuccessful 3 times (3 x 10 seconds = 30 seconds).

   1.5. Requirement:
   Connection-retries must be sent every 10 seconds.

   1.6. Requirement:
   When the connection is established initially and the devices are communicating, the window must be successfully opened and closed once as a mechanical functionality test.

2. Requirement: Invalid Value Handling

   2.1. Requirement:
   If the measured value differentiates more than 50 ppm from the average of the 5 previous values, the value is marked as invalid.

   2.2. Requirement:
   If the sensor value is marked as invalid, the value must be dropped.

3. Requirement: Communication between devices

    3.1. Requirement:
    For every message sent between the Raspberry Pi and the Arduino, the receiving device must send an acknowledgement-message to the original sender.

    3.2. Requirement:
    If the Arduino doesn't get an acknowledgement-message for one of his status messages, it must resend this message.

    3.3. Requirement:
    Not acknowledged resend-messages must be sent every 10 seconds.

    3.4. Requirement:
    If 1 following resend-messages are not acknowledged, an LED on the Arduino must start blinking.

4. Requirement: Operating the window

    4.1. Requirement:
    It must be checked whether the window has been opened or closed sufficiently, by comparing the measured end-position of the stepper motor with the value of a fully opened or closed window

    4.2. Requirement:
    If the measured end-position of the stepper motor deviates by 1 cm from the value of a fully opened window when opening, the motor must stop immediately and go back into its starting position.

    4.3. Requirement:
    If the measured end-position of the stepper motor deviates by 1 cm from the value of a fully closed window when closing, the motor must stop immediately and go back into its starting position.

    4.4. Requirement:
    After the stepper motor has to reset to his starting position when opening or closing the window, it must retry the opening or closing process a second time after 10 seconds.

    4.5. Requirement:
    If the opening or closing process of the windows doesn't work a second time an error has to be displayed by the Raspberry Pi through an blinking LED.

6.2 Security related requirements

1. Spoofing Requirement:

   1.1. Requirement:
   Both devices (Arduino and Raspberry Pi) must share a pre-shared key (PSK), which is used for authentication and integrity verification.

   1.2. Requirement:
   For each communication, the sender must compute an HMAC using the message and the secret key, which is validated by the receiver using the same key.

   1.3. Requirement:
   To exchange the key, both devices must use the Diffie-Hellman key exchange.

   1.4. Requirement:
   Both devices must have a unique device ID that is checked to ensure that only authenticated devices are allowed to communicate.

2. Tampering Requirement:

   2.1. Requirement:
   Each message between the Arduino and Raspberry Pi must be accompanied by an HMAC to verify whether the message has been altered during transmission.

   2.2. Requirement:
   Unauthorized physical access of the Raspberry Pi or Arduino should be prevented.

3. Repudiation Requirement:

   3.1. Requirement:
   Each sent and received packet must be logged with timestamps on the Arduino and the Raspberry Pi.

   3.2. Requirement:
   Critical actions, such as opening and closing the windows, must also be logged.

4. Information Disclosure Requirement:

4.1. Requirement:
The communication over the Bluetooth connection between the Raspberry Pi and the arduino must be encrypted via AES-CBC-128.

4.2. Requirement:
The successful encryption must be verified, by sending an encrypted default string with the data for the devices to verify the encryption.

5. Denial of Service Requirement:

5.1. Requirement:
The Raspberry Pi must accept no more than 1 message every 9 seconds from the Arduino to prevent flooding attacks.

5.2. Requirement:
The Arduino must only wait for 1 acknowledgement message after sending a data message to the Raspberry Pi.

5.3. Requirement:
The Arduino must ignore all other received acknowledgement messages.

6. Elevation of Privilege Requirement:

6.1. Requirement:
There is a minimum length of 12 characters for all used passwords.

6.3 Requirements with no influence on Safety and Security

1. Requirement:
The Raspberry Pi must be waiting for the Arduino to establish a Bluetooth Connection.

2. Requirement:
The Adruino must try to initiate a Bluetooth Connection to the corresponding Raspberry Pi.

3. Requirement:
If Bluetooth communication fails (no acknowledgment received), the Raspberry Pi must attempt to reconnect up to 3 times.

4. Requirement:
   If the reconnection fails, a visual signal (the Error-LED starts blinking) will be triggered, and the system will wait 10 seconds before retrying.

5. Requirement:
   The Arduino with the CO2 sensor must measure the current CO2 levels every 10 seconds and evaluate the value.

6. Requirement:
   The Raspberry Pi must control the motor to open and close the windows and switches the fan on or off based on the sent status of the Arduino.

7. Requirement:
   The Arduino must communicate the current status to the Raspberry Pi (OPEN, CLOSE, IDLE).

8. Requirement:
   When instructed by the Arduino via receiving an OPEN-Status, the Raspberry Pi must control the motor to open the window, start the fan and enter the Alarm state. This is triggered after measuring 10 values that are 5 ppm higher than the average.

9. Requirement:
   When instructed by the Arduino via receiving a CLOSE-Status, the Raspberry Pi must must control the motor to close the window, stop the fan and revert back to the Idle state. This happens when 10 values are measured that are within 2 ppm of the average.

10. Requirement:
    The project does not need to support any features not mentioned.