Design document

Group Nr. 3

Timur Ali Basnakajev
Tobias König
Jessica Marban
Patrick Neumann

Contents

1 Project description

1.1 Problem desription

When the CO2 level in a room reaches a certain level, a window should automatically open and a fresh air fan should switch on. After a certain lower CO2 level is reached and a certain run-on time has elapsed, the window is closed again and the fan is switched off again.

1.2 Hardware

1. MH-Z19 CO2 Sensor
2. Arduino with bluetooth module
3. Raspberry
4. Electric motor

1.3 Hardware

1. Analog input for measuring the CO2 contentr
2. Digital output for controlling a window opener
3. Digital output to control a fan

1.4 Schema

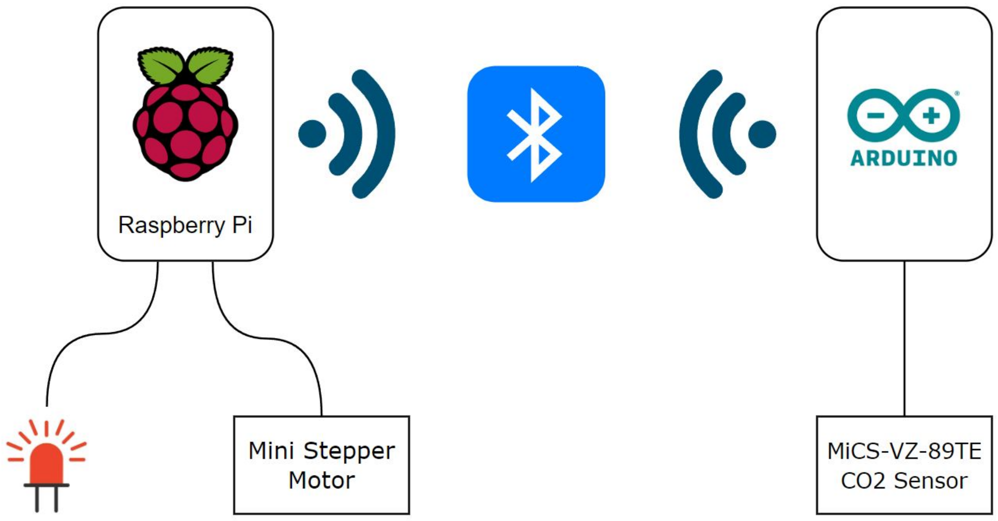

Figure 1: system description

2 Software architecture and design

2.1 Software modules

Safety related modules

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

   - Receive status messages (e.g., OPEN, CLOSE, IDLE) from the Arduino.
   - Open/close the window using the stepper motor based on the status message.
   - Monitor motor movement and detect deviations during window operation.
   - Stop motor and trigger an error state if obstacles are detected.
   - Report errors through email and LED indicators.

   Data: Stepper motor position, window state, status messages.
   Requirements: 4. (Operating the window), 1.5. (Connection retries and email alerts), 3.1. (Acknowledgment of messages).

2. **Value Validation Module (Arduino):**
   Description: This module validates the CO2 sensor readings and determines whether the values are within a valid range. Only the status (OPEN, CLOSE, IDLE) is sent to the Raspberry Pi based on the results.
   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.

   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.

   Data: Encrypted communication data, status messages, acknowledgment messages.
   Requirements: 3. (Communication between devices), 1. (Spoofing and key exchange), 4. (Information Disclosure), 1.5. (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 messages received from the Arduino. It turns the fan on when instructed to open the window and turns it off when instructed to close the window.
   Functions:

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

   Data: Status messages, 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:

- Monitor system health, such as CPU temperature and memory usage.
- Provide status updates about system performance.
- 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:

- **MH-Z19 Library (Arduino):** ‘MHZ19uart‘ 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:**
  - **Raspberry Pi** and **Arduino**: AES libraries in C for 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 obstacle detection during motor operation to prevent damage.
- **Priority 3:** Status validation on the Arduino to ensure accurate CO2 evaluation.

2.4 Pinout

- **CO2 Sensor (MH-Z19 - 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):** LED connected to GPIO pins for status indication.

3 Program flowchart

IDP_Designdokument/images/ablaufdiag_pi.drawio.png

Figure 2: Rasperry PI

IDP_Designdokument/images/ablaufdiag_adriuno.drawio.png

Figure 3: Arduino

4 Hazard identification

4.1 Identified hazards and countermeasures

1. Hazard 1:

   Co2 sensor delivers incorrect values 2.

2. Hazard 2:

   The window is stuck and does not open or close 3.

3. Hazard 3:

   Arduino fails 4.

4.2 Identified hazards without countermeasures

1. Hazard 1:

   The Co2 sensor fails and does not provide data

2. Hazard 2:

   The power supply fails

3. Hazard 3:

   The engine fails

5 Threat identification

5.1 Identified threats and countermeasures

1. **Threat 1:** Someone could manipulate the window this way, intentionally or unintentionally that it no longer closes after opening. This could result in a break-in consequences.

   Mitigation: The device carries out a self-test when commissioning (open 1 x and close 1 x)

2. **Threat 2**: The window couldn't open because someone put something in front of the window. This means there is no ventilation. The system no longer works.

   Mitigation: Refer to mitigation of Threat 1:

3. **Threat 3**: DoS: Someone could carry out a DoS attack on the Rasperry PI. This will prevents emails from being sent.

   Mitigation: Installation of a firewall. This means that the Rasperry PI cannot be reached from outside.

5.2 Identified threats without countermeasures

1. **Threat 1:** Someone could try to deliberately trigger the CO 2 sensor. This will open the window opened and a break-in could be carried out through the window.
2. **Threat 2:** Someone could install a jammer and thus disrupt the connection between Rasperry PI and Arduino and thus, for example, prevent the window from closing to carry out a break-in.
3. **Threat 3:** Someone could install a jammer and prevent the safety circuit from tripping. This means the system no longer works.

6 Requirements

6.1 Safety related requirements

1. Requirement: Inital Connection Handling

    1.1. Requirement:
    If an error occurs during program start at the Arduino, this must be indicated by an LED (LED OFF)

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

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

    1.4. Requirement:
    If an error occurs during program start at the Raspberry Pi, this must be indicated by an LED (LED OFF)

    1.5. 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 for 30 seconds.

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

    1.7. 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:
    Every measured value of the CO2 sensor must be checked for their validity by the Arduino.

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

2.3. 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:
   Both devices (Arduino and Raspberry Pi) must share a pre-shared key (PSK), which is used for authentication and integrity verification. For each communication, the sender computes an HMAC using the message and the secret key. The receiver uses the same key to validate the HMAC.
   To exchange the key, both devices must use the Diffie-Hellman key exchange.
   Both devices must have a unique device ID that is checked to ensure that only authenticated devices are allowed to communicate.

2. Tampering 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.
   Unauthorized physical access of the Raspberry Pi or Arduino should be prevented.

3. Repudiation Requirement:
   Each sent and received packet must be logged with timestamps on the Arduino and the Raspberry Pi.
   Critical actions, such as opening and closing the windows, must also be logged.

4. Information Disclosure Requirement:
   The communication over the Bluetooth connection between the Raspberry Pi and the arduino must be encrypted via AES-CBC-128.
   The successful encryption must be verified. This could be done by sending an encrypted default string with the data for the devices to verify the encryption.

5. Denial of Service Requirement:
   The Raspberry Pi must accept no more than 1 message every 9 seconds from the Arduino to prevent flooding attacks.
   The Arduino must only wait for 1 acknowledgement message after sending a data message to the Raspberry Pi.

The Arduino must ignore all other received acknowledgement messages.

6. Elevation of Privilege Requirement:
   There is a minimum length for the password of 12 characters. This guarantees a strong password which will make it hard for attackers to take over control of the administrators account.

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 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 (blinking LED) 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 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 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.