FH CAMPUS WIEN
UNIVERSITY OF APPLIED SCIENCES

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

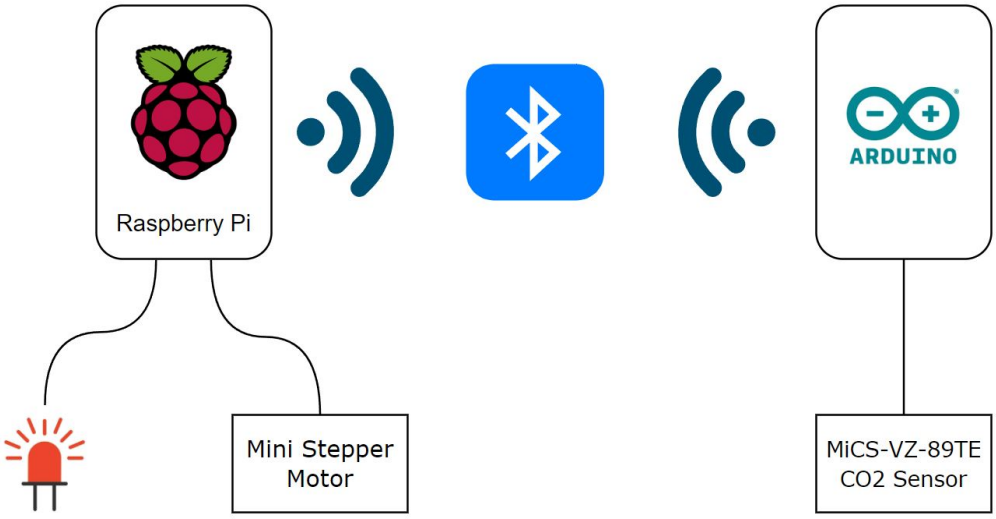

Figure 1: system description

2 Software architecture and design

2.1 Software modules

Safety related modules

1. **CO2 Sensor Measurement and Validation Module (Arduino):**
   Description: This module reads CO2 concentration levels using the MH-Z19 sensor and validates the readings to ensure they fall within a typical range. If a reading is outside the acceptable range, the value is marked as invalid and appropriate actions are taken. Valid CO2 values are sent to the Raspberry Pi for further processing.
   Functions:

   - Measure CO2 concentration as an analog value using the MH-Z19 sensor.
   - Validate the analog reading to ensure it falls within a typical range.
   - Mark and handle invalid values (e.g., send an error message or retry reading).
   - Send validated CO2 concentration values to the Raspberry Pi via serial communication.

   Data: Raw sensor data, validated CO2 concentration values (ppm).
   Requirements: 2., 2.2., **??**

2. **Window Control Module (Raspberry Pi):**
   Description: This module receives validated CO2 values from the Arduino and controls the electric motor to open or close the window based on the CO2 levels. It ensures safe operation by detecting any obstacles during movement.
   Functions:

   - Receive validated CO2 concentration values from the Arduino.
   - Open/close the window using the stepper motor if CO2 levels exceed or drop below predefined thresholds.
   - 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, validated CO2 data.
   Requirements: 3., 3.1., 3.2., 3.3.

3. **Fan Control Module (Raspberry Pi):**
   Description: This module controls the fan based on the CO2 levels received from the Arduino. It turns the fan on when CO2 levels are high and turns it off when levels normalize.
   Functions:

   - Turn on the fan when the CO2 level surpasses a certain threshold.

- Turn off the fan when CO2 levels fall below the desired range.

Data: CO2 concentration values, fan state.
Requirements: Related to system operation based on CO2 levels.

Security related modules

1. **Communication Encryption Module:**
   Description: This module ensures secure data transmission between the Raspberry Pi and Arduino to prevent unauthorized access and tampering. It requires both the Raspberry Pi and Arduino to handle encryption and decryption of messages.
   Functions:

   - On the Raspberry Pi, encrypt data using the 'cryptography' library before sending it to the Arduino.
   - On the Arduino, use an AES-compatible library (e.g., Arduino Cryptography Library) to decrypt the received data.
   - Ensure both devices securely share the encryption key for symmetric encryption.

   Data: Encrypted communication data.
   Requirements: 1.

2. **System Hardening Module (Raspberry Pi):**
   Description: This module focuses on securing the Raspberry Pi against cyber threats. It ensures that the Raspberry Pi is protected from unauthorized access.
   Functions:

   - Manage user authentication and access control.
   - Implement firewall rules to restrict network access.
   - Monitor and log suspicious activity.

   Data: Security logs, access control records.
   Requirements: 2.

Modules with no influence on Safety and Security

1. **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..

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.
  - **Raspberry Pi**: ‘pyserial‘ - for reading sensor data sent by the Arduino.
- **Stepper Motor Control Library:**
  - **Raspberry Pi**: ‘RPi.GPIO‘ or ‘gpiozero‘ - to control the stepper motor for window movement.
- **Email Notification Library:** ‘smtplib‘ (Python) - for sending email alerts directly from the Raspberry Pi when error states are detected.
- **Encryption Libraries:**
  - **Raspberry Pi**: ‘cryptography‘ (Python) - for encrypting communication before sending it to the Arduino.
  - **Arduino**: ‘Arduino Cryptography Library‘ - for decrypting the received encrypted messages.
- **Firewall Configuration Tools:** ‘ufw‘ (Linux tool) - for setting up and managing firewall rules on the Raspberry Pi.

2.3 Interrupts

Definition of priorities:

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

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

Hier bitte Sequenzdiagramme, bzw. Programmablaufdiagramme

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:
   At program start all safety related functions must be tested.

   1.1. Requirement:
   If an error occurs during program start, this must be indicated by an LED (flashing at 1 second intervals)

   1.2. Requirement:
   Communication between Raspberry and Arduino must be established.

   1.3. Requirement:
   When the system starts, the window must be successfully opened and closed once.

2. Requirement:
   The values of the CO2 sensor must be continuously checked for their validity 1:

   2.1. Requirement:
   If the sensor value is outside the typical range, the value must be marked as invalid.

   2.2. Requirement:
   If the sensor values are invalid an email must be sent.

3. Requirement:
   It must be checked whether the window has been opened or closed sufficiently 2:

   3.1. Requirement:
   If the measured position of the stepper motor deviates by 5 mm when opening, the motor must stop immediately and go into error state (obstacle was detected).

   3.2. Requirement:
   If the measured position of the stepper motor deviates by 5 mm when closing, the motor must stop immediately and the window must open again (obstacle has been detected).

3.3. Requirement:
If an obstacle is detected during opening or closing, the system must report an error (email and LED).

4. Requirement:
It must be checked whether the Arduino is working properly 1:

4.1. Requirement:
The communication between Raspberry and Arduino must be checked cyclically (10 second interval).

4.2. Requirement:
If the communication between the Raspberry and the Arduino is interrupted for more than 30 seconds, the system must go into error state (Email and LED).

6.2 Security related requirements

1. Requirement:
The communication between the Raspberry and the Arduino must be encrypted.

2. Requirement:
The Raspberry PI must be hardened against cybersecurity attacks.

6.3 Requirements with no influence on Safety and Security

1. Requirement:
The system should be installed in an inaccessible location.