Report: Simple intruder alarm system - Embedded systems project 2024

General Description

This motion detection alarm system allows arming and disarming the system by means of a password or a contactless card (RFID). The system includes a database for card management, where new cards can be registered, existing cards can be deleted or blocked. Card management can only be performed using a "MASTER" card, defined during system initialization. The following describes the operation of the system divided into three main states: Initialization, System Disarmed and System Armed.

Status: Initialization

In the initialization state, the "MASTER" card and the password that will allow the management of the system are configured. This process is carried out when the system is started for the first time.

INITIALIZATION PROCEDURE:

- **Definition of the "MASTER" card:**
  - The "MASTER" card is initially registered and will be the only one with privileges to manage other cards (add, delete or block).
  - A specific RFID card is scanned and its ID is stored as the "MASTER".
  - Use of the function "initialize_Master_card()" which uses the function "read_card_info()" that allows us to read the information of a card on the reading device.
  - The information will be stored in the database by the "add_card_db(card_id, auth_card_id)" method.

- **Password Configuration:**
  - The system prompts for an initial password that will be used to arm and disarm the system.
  - The password is entered through the keyboard (keypad 4x4) and saved in the system.
  - Use of the "define_pass()" function which, by detecting the buttons typed on the keyboard, allows you to define a password.

At the end of the initialization, we would enter the "System Disarmed" and "System Armed" states, from which the system will not exit until a reboot of the raspberri is performed. This is because the main operation of the system is in these two states that will be interchanged. There is no need to go back to initialization, allowing us to create a more realistic system.

Status: System Disarmed

In the disarmed state, the system allows card management, log display and the option to put the device in "sleep" mode. These options will be displayed on the LCD, where we will be able to configure the device by using the keypad using the "scan_keypad()" function. The button used will be analyzed and processed to lead to one functionality or another.

DISASSEMBLED SYSTEM FUNCTIONALITIES:

- **Card Management "1:CardM" function "card_manager_assistant(master)":**
    - After entering this configuration, we will be asked for the "MASTER" card, in order to access the "card_manager_assistant(master)" function. By means of it, we will be able to perform these four functionalities:
        - **Add Card "1ADD":** Only the "MASTER" card can register new cards using the function "add_card_db(card_id, auth_card_id)".
        - **Block "2BLK" card:** The "MASTER" card can block registered cards using the "block_card(card_id,auth_card_id)" function and only if the card has been previously registered. It will not be possible to block the "MASTER" card for system integrity and consistency reasons.
        - **Delete Card "3:DLT":** The "MASTER" card can remove previously registered cards using the function "remove_card(card_id, auth_card_id)" and only if the card has been previously registered. The "MASTER" card cannot be blocked for system integrity and availability reasons.
        - **Show card DB "4:RG":** By means of the "print_db_file()" function, the "card_db.txt" file will be shown where the database of the registered cards and their status will be displayed. With it, we will be shown a timestamp that shows the last modification of this file.

If and when a card id is required, the system will display a text on the LCD after selecting the function indicating when the new card must be inserted in order to be read. In addition, in the case where the card that has been set is allowed, an error will be displayed and logged. In the opposite case, it will indicate that the procedure has been done correctly by means of a "Successful" on the LCD. This "Card Management" mode will only be maintained for 30 seconds.

- **Arm System "2:armSy".**
    - Changes the "systemStatus" parameter to "armed", allowing us to switch to the "System Armed" status.
- **Display of the Log File "3:LR":**
    - The activity logs are stored with timestamps and can be visualized through the "print_log_file()" function that will show all the movements of the device after it has been started. This will occur after having made a copy of the log file by means of the "create_backup()" method, which will allow us to have a history of

"backups" with a timestamp in the file name inside the "backup" folder ("/sd/backup/log_backup_{timestamp}.txt").

- **Sleep Mode "P:Slp":**
    - The system will simulate a "sleep" mode by means of the "system_off()" function, allowing us to see the current time only on the LDC. We will be able to exit this mode and return to the "System disarmed" state by pressing the "Power" P button again.

Status: Armed System

In the armed state, the PIR sensor is active and detects motion. The system can only be disarmed with the password or an active card.

In the armed state, the PIR sensor is activated, that means that the variable "sensor_pin", which refers to the pin of the PIR sensor, will have an interrupt defined with a handler to the function "buzzer_interrupt_pswd()". This is done by the function "enable_interruption()", in the same way that "disable_interrupt()" disables the interruption. In this way, the PIR will only operate the interruption routine when we are in the "System Armed" state.

FUNCTIONALITIES OF THE ASSEMBLED SYSTEM:

- **Motion Detection:**
    - If the PIR sensor detects motion, the buzzer is activated.
- **Sensor disconnection:**
    - If the sensor is disconnected, the buzzer is also activated.
- **Interrupt Routine ""buzzer_interrupt_pswd()":**
    - It can only be deactivated with an active card within 30 seconds, allowing us to return to the "System Disarmed" status. In case a card with "active" status is detected, the "systemStatus" variable will change to "disarmed" and we will return to the "System Disarmed" status.
    - As long as we remain within the 30 seconds, the function will continuously call the function "play_tone(frequency, single_duration)" which will activate and deactivate the buzzer to make it play at a given frequency for a given duration.
    - If any card is displayed that has not been registered or has been blocked, a message will be displayed on the LCD and recorded in the log file with a differentiated mark.
    - If an active card is not presented within 30 seconds, the system simulates an "emergency call" and locks in a loop.

Modules

- `rfid_lector.py`: Management of RFID cards and reading.
- `keypad_4x4.py`: Reading data from the keyboard.
- `rtc_sdCard.py`: Creation of timestamps and management of the SD card.
- `LCD.py`: Control of the LCD screen.

Libraries folder "lib"

- `ds3231.py`: Library for the RTC.
- `lcd_api.py`: API for the LCD screen.
- `mfrc522.py`:Library for the RFID reader.
- `pico_i2c_lcd.py`: Control of the LCD screen by I2C.
- `sdcard.py`: Handling the SD card.

Components

- Raspberry Pi Pico
- RFID Module MFRC522
- RTC DS3231
- 16x2 I2C LCD screen
- Keypad 4x4
- PIR sensor
- Buzzer (embedded)
- SdCard

Connexions

• **RFID MFRC522**: ○ **SDA**: GP22 ○ **SCK**: GP2 ○ **MOSI**: GP3 ○ **MISO**: GP0 ○ **IRQ**: Not Connected ○ **GND**: GND ○ **RST**: GP21 ○ **VDC**: 3.3V	• **Keypad 4x4**: ○ **R1**: GP9 ○ **R2**: GP8 ○ **R3**: GP7 ○ **R4**: GP6 ○ **C1**: GP1 ○ **C2**: GP13 ○ **C3**: GP4 ○ **C4**: GP5
• **I2C LCD**: ○ **SDA**: GP16 ○ **SCL**: GP17 ○ **VDC**: 5V ○ **GND**: GND	• **RTC DS3231**: ○ **SDA**: GP26 ○ **SCL**: GP27 ○ **VDC**: 3.3V ○ **GND**: GND
• **Buzzer**: ○ **OUT**: GP18	• **PIR sensor** ○ **VDC**: 3.3V ○ **IN:** GP20 ○ **GND**: GND
• **SdCard** ○ **SDA**: GP15 ○ **SCK**: GP10 ○ **MOSI**: GP11 ○ **MISO**: GP12	

Conclusion

This project integrates several electronic components to create a functional alarm system that logs and displays real-time access using RFID technology and a real-time clock. The addition of the matrix keypad allows for additional interaction with the system, offering greater flexibility and accessibility. As a proposal to improve the system we could implement the use of a second processor. In this way, we could fix some problems encountered when implementing card or button detection functions that maintain an active wait and slow down the system performance noticeably. This situation is more present when having active wait functions together within the same loop.