Sistemi Embedded e Real Time

Home Alarm System su Esp 32

Pietro Orlandi - matricola 161052

Andrea Montanari - matricola 162639

DESCRIZIONE

In questo progetto si è voluto realizzare un prototipo di un sistema di allarme domestico. Nello specifico si simula un sistema di allarme per un appartamento con tre stanze.

Per realizzare l'obiettivo si sono utilizzati vari componenti hardware:

- un tastierino fisico che permette di prendere come input il PIN del sistema e nel caso sia corretto permette di spegnere o accendere l'allarme;
- un servo a 180 gradi che simula una videocamera con possibilità di rotazione. L'idea è quella che inizialmente la videocamera sia a 90 gradi, e quando rileva un movimento in una delle stanze essa si ruoti in modo da riuscire a "filmare" cosa sta accadendo nella stanza in cui si è verificata l'intrusione;
- un sensore per ogni stanza con cui è possibile verificare se c'è stata un'intrusione, più nello specifico:
    - nella stanza 1 è presente un PIR (sensori di movimento a infrarossi hc-sr501) per rilevare i movimenti;
    - nella stanza 2 è presente un PIR per rilevare i movimenti;
    - nella stanza 3 si è voluto simulare un sensore che rilevi l'apertura di una finestra. Tale simulazione è realizzata con un un bottone presente sulla breadboard: il click del bottone corrisponde all'apertura della finestra.
- Un buzzer che suona nel caso venga rilevata un'intrusione;
- un LED RGB che permette di avere un feedback visivo riguardo allo stato dell'allarme (l'allarme può essere nello stato OFF, ON e TRIGGERED, corrispondenti ai colori blu, verde e rosso).

Oltre all'allarme fisico descritto sopra, è possibile collegare l'allarme tramite l'applicazione Android e IoS Blynk IoT.

Questo ci permette di avere informazioni in tempo reale sul nostro smartphone riguardo le possibili intrusioni nelle varie stanze e sullo stato dell'allarme. Inoltre, quando viene inserito il PIN dall'utente, è presente un LCD virtuale che ci permette di avere un feedback visivo. Infine, tramite l'app è possibile modificare la rotazione della videocamera (anche da allarme spento) per permettere ad un utente di monitorare gli ambienti di casa.

SOFTWARE TOOLS

Gli strumenti software utilizzati per la realizzazione del progetto sono:

- Visual Studio Code con l'estensione per Arduino di Microsoft per la scrittura del codice, la compilazione e il caricamento del programma sulla scheda ESP32;
- La libreria "esp32" di Espressif per Arduino, la quale include FreeRTOS;
- Git e GitHub per la condivisione di codice.

TASKS

Di seguito sono elencati i task dell'applicazione e le relative funzioni.

- **TaskPin**: controlla periodicamente l'inserimento di un valore sul tastierino numerico e nel caso sveglia il TaskStamp per visualizzare il codice.
- **TaskStamp**: viene sbloccato dal TaskPin quando viene inserito un carattere, si occupa di stampare i valori su LCD virtuale di Blynk (e nel seriale). Quando sono stati inseriti 4 caratteri (quindi è un possibile pin di accesso), viene controllato se corrisponde al pin di sistema. Nel caso affermativo si occupa di cambiare lo stato dell'allarme (OFF -> ON, ON -> OFF o TRIGGERED -> OFF). Inoltre sveglia i task a seconda della casistica attraverso la xSemaphoreGive.
- **TaskMotionSensor**: inizialmente siccome l'allarme sarà OFF verrà bloccato sul suo semaforo privato. Quando poi l'allarme andrà nello stato ON, verrà sbloccato dal TaskStamp e controllerà periodicamente se il PIR associato ha rilevato movimenti.
  Quando viene rilevato movimento si distinguono due casi:
  - lo stato dell'allarme non è in TRIGGERED: verrà cambiato lo stato dell'allarme in TRIGGERED e saranno svegliati i task della sirena, del LED e del servo (in modo da ruotare la videocamera) attraverso la xSemaphoreGive sui relativi semafori privati;
  - lo stato dell'allarme è già TRIGGERED: verrà svegliato il servo per effettuare una rotazione della videocamera verso la propria posizione.
- **TaskWindowSensor**: il comportamento è pressoché identico al TaskMotionSensor ma con il controllo effettuato su un bottone (che, ricordiamo, simula l'apertura di una finestra).
- **TaskSiren**: inizialmente è bloccato nel suo semaforo privato, viene poi svegliato quando per la prima volta l'allarme passa da ON a TRIGGERED indicando che è stata rilevata un'intrusione. In questo caso viene attivato il buzzer.
  Si risveglia anche quando l'allarme passa da TRIGGERED a OFF, caso in cui il buzzer viene disattivato.
- **TaskServo**: inizialmente è bloccato nel suo semaforo privato, viene poi svegliato quando un sensore rileva un movimento.
  Viene svegliato anche quando si modifica la rotazione della videocamera da app Blynk attraverso lo slider.
  Una volta attivo, la sua funzione è quella di ruotare il servomotore verso la stanza in cui il sensore è stato attivato oppure nella direzione indicata dall'utente tramite lo slider nell'app Blynk.
- **TaskLed**: il colore del LED rappresenta lo stato dell'allarme. Inizialmente questo task è bloccato nel suo semaforo privato, viene poi svegliato quando si verifica un cambio di stato dell'allarme. Può quindi venire svegliato da TaskStamp (quando l'utente inserisce il PIN corretto e allarme va OFF o ON), oppure dai task dei sensori (quando rilevano il movimento la prima volta e l'allarme passa da ON a TRIGGERED).
  Un'altra funzione è quella di settare i colori dei led virtuali dell'app Blynk.
- **TaskBlynk**: è il task che permette di comunicare con l'app Blynk e di conseguenza interfacciare l'allarme anche da telefono. Esegue periodicamente la Blynk.run() che si occupa di mantenere la connessione con il server Blynk e di mandare e ricevere i dati.

Semafori e mutex

Di seguito sono elencati i semafori e i mutex utilizzati per gestire la sincronizzazione e la mutua esclusione.

- Mutex: usato per proteggere l'accesso alle risorse condivise (come lo stato dell'allarme e la posizione del servo) .
- s_stamp: semaforo privato del TaskStamp.
- s_sensor: semaforo privato N-Ario (FreeRTOS Counting Semaphores) con cui vengono bloccati e svegliati i task che gestiscono i sensori di movimento e il bottone che simula l'apertura della finestra.
  Siccome i due TaskMotionSensor a cui sono associati i 2 PIR e il TaskWindowSensor, a cui è associato il bottone, hanno lo stesso comportamento (devono bloccarsi all'inizio, devono essere svegliati dal TaskStamp quando allarme diventa ON), è opportuno che essi utilizzino lo stesso semaforo. A supporto di questo semaforo è presente un contatore dei bloccati.
- s_siren: semaforo privato del TaskSiren.
- s_LED: semaforo privato del TaskLed.
- s_servo: semaforo privato del TaskServo.

Requisiti

I requisiti dell'applicazione sono stati soddisfatti nel seguente modo:

- ≥ 3 canali di input: tastierino a membrana, 2 sensori PIR, 1 bottone.
- ≥ 2 sensori oppure 1 sensore + 1 dispositivo di rete: 2 sensori PIR, Wi-Fi integrato nella scheda ESP32;
- ≥ 2 canali di output: buzzer, LCD virtuale, servo, LED RGB;
- ≥ 1 task per comunicazione molti-a-uno: i task dei sensori di movimento e della finestra comunicano i cambiamenti di stato al task stamp e al task Blynk, a quest'ultimo vengono comunicati anche i movimenti rilevati.

Requisiti addizionali

Tra i requisiti addizionali:

- Git versioning: utilizzo di Git e Github per la condivisione di codice.
- Periodic execution (timer-based tasks): esecuzione periodica dei task.
- Footprint minimization.
- MISRA C compliance: verifica dell'adempimento alle linee guida di sviluppo software nel contesto dei sistemi embedded tramite l'utilizzo del tool di analisi *cppcheck*.