ASSIGNMENT #2 - SMART BRIDGE

Embedded Systems and Internet of Things

A.A. 2022-2023

Ugo Baroncini, Luca Bighini, Pietro Tellarini

Project Demonstration Video

Descrizione

Il prototipo ha intenzione di simulare un sistema embedded di un ponte che svolge le seguenti funzioni:

- monitoraggio del livello dell'acqua e in caso di allarme la conseguente apertura di una valvola per il deflusso.
- un lampione automatico si accende o spegne in base ai sensori di luminosità e di movimento.

Suddivisione in stati

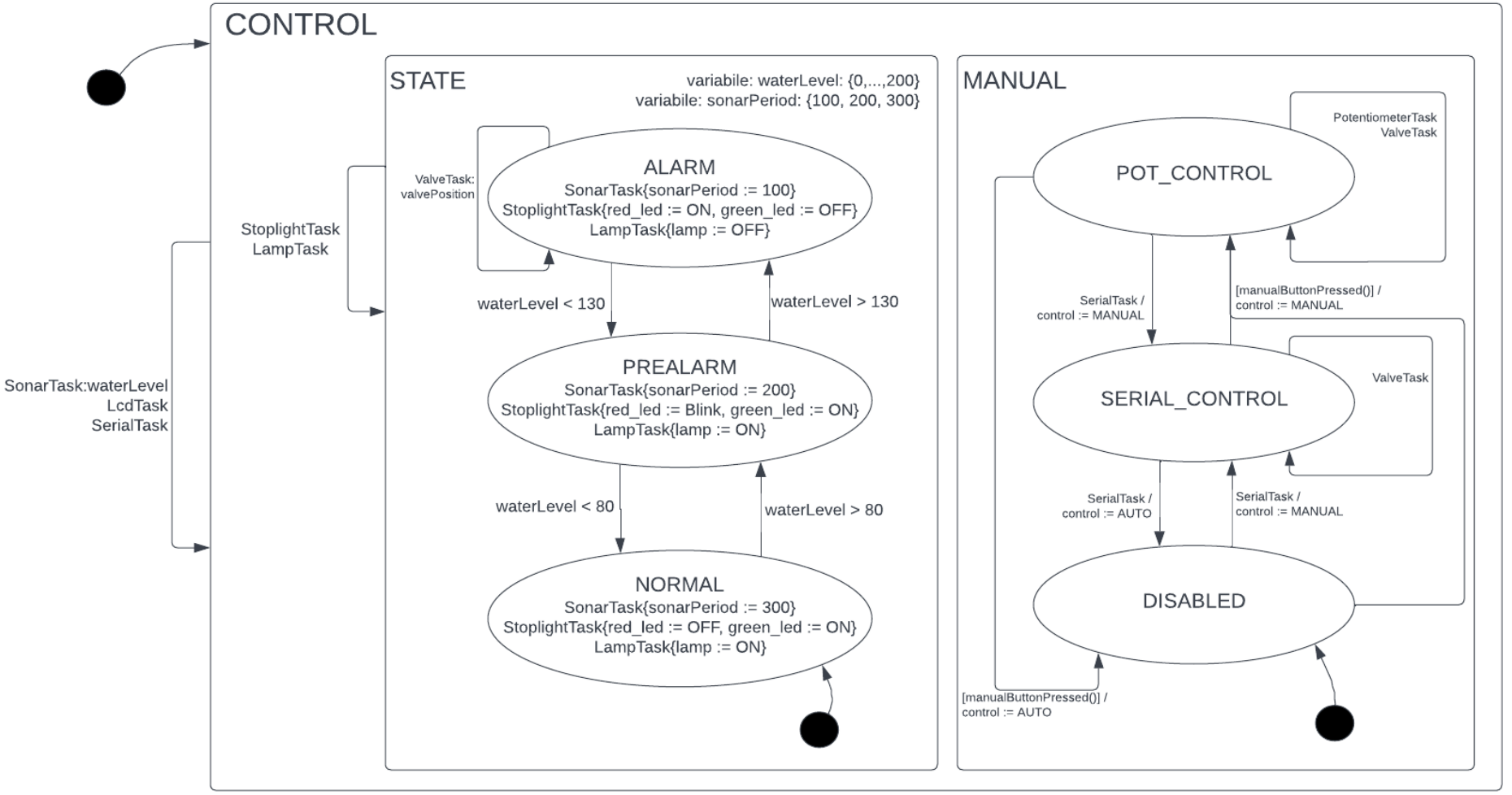
Per rappresentare al meglio tale sistema di controllo si è deciso di procedere con una suddivisione in stati, che lavorano parallelamente. Lo Stato (path: "lib\Status\src") aggiornato del sistema viene istanziato nel main e passato come parametro ai task.

Nello Stato sono presenti i possibili stati del sistema:

- **Control** {MANUAL, AUTO}
  Il Control indica se la valvola per il deflusso dell'acqua viene azionata in modo manuale o automatico. Il sistema inizialmente in modalità automatica può passare in modalità di controllo manuale solo se si è in stato di allarme. Se invece il sistema si trova in manuale può tornare in automatico in qualsiasi momento.
- **State** {NORMAL, PREALARM, ALARM}.
  Lo State indica le tre fasi in cui si può trovare il ponte in base al livello dell'acqua; al crescere del livello dell'acqua gli stati passano da NORMAL a PREALARM a ALARM.
- **ManualControlSource** {DISABLE, POT_CONTROL, SERIAL_CONTROL}
  Il ManualControlSource indica se la sorgente dei dati per il controllo manuale è attiva e proviene dal potenziometro o dalla porta seriale. Se il controllo è manuale si può sempre passare dal controllo tramite il potenziometro a quello seriale.

Nello stato sono mantenute ulteriori informazioni utili:

- **waterLevel**: livello corrente dell'acqua misurato dal Sonar.
- **valvePosition**: angolo corrente di apertura della valvola.
- **SerialValvePosition**: angolo di apertura della valvola impostato dalla seriale.
- **PotValvePosition**: angolo di apertura della valvola impostato dal potenziometro.

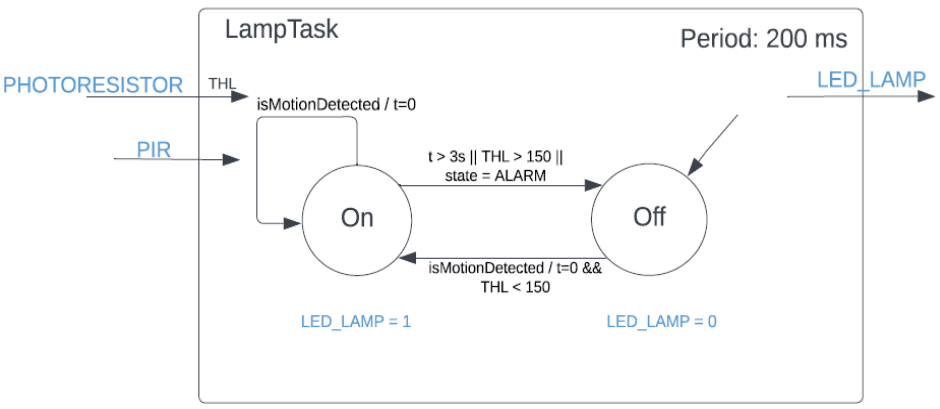

CONTROL
STATE
variabile: waterLevel: {0,...,200}
variabile: sonarPeriod: {100, 200, 300}
ALARM
SonarTask{sonarPeriod := 100}
StoplightTask{red_led := ON, green_led := OFF}
LampTask{lamp := OFF}
ValveTask:
valvePosition
StoplightTask
LampTask
waterLevel < 130
waterLevel > 130
PREALARM
SonarTask{sonarPeriod := 200}
StoplightTask{red_led := Blink, green_led := ON}
LampTask{lamp := ON}
SonarTask:waterLevel
LcdTask
SerialTask
waterLevel < 80
waterLevel > 80
NORMAL
SonarTask{sonarPeriod := 300}
StoplightTask{red_led := OFF, green_led := ON}
LampTask{lamp := ON}
MANUAL
POT_CONTROL
PotentiometerTask
ValveTask
SerialTask /
control := MANUAL
[manualButtonPressed()] /
control := MANUAL
SERIAL_CONTROL
ValveTask
SerialTask /
control := AUTO
SerialTask /
control := MANUAL
DISABLED
[manualButtonPressed()] /
control := AUTO

Suddivisione in Task

Il sistema di controllo Smart Bridge prevede inoltre l'utilizzo di Task per gestire tutte le componenti e i loro comportamenti. Le task sotto elencate hanno funzioni specifiche e si occupano, eventualmente, di modificare lo Stato condiviso, in base alle richieste del problema. Tali task vengono istanziate e inizializzate nella fase di setUp del main e inserite nello scheduler che si occuperà di eseguirle periodicamente.

Lo scheduler esegue i seguenti task:

- **LampTask**
  si occupa di gestire l'accensione e lo spegnimento del lampione (Led). L'accensione avviene solo in caso di scarsa luminosità (Photoresistor) e in presenza di un movimento (Pir). Il lampione rimane acceso per 3 secondi dopo l'ultimo movimento rilevato e si spegne immediatamente in stato di ALARM.
- **LcdTask**
  si occupa di mostrare a video (LCD) le informazioni riguardanti il sistema in base alla stato in cui esso si trova.
- **PotentiometerTask**
  si occupa di leggere i valori restituiti dal potenziometro (Potentiometer) e salvarli nella relativa variabile all'interno dello Stato.
- **SerialTask**
  si occupa di mandare e leggere dati dalla seriale modificando se necessario le relative variabili dello Stato.
- **SonarTask**
  si occupa di leggere i valori restituiti dal sonar (Sonar) e salvarli nella relativa variabile all'interno dello Stato. Tale task è in grado di cambiare il proprio periodo di esecuzione se avviene un cambiamento dello State.
- **StoplightTask**
  si occupa di dell'accensione dei semafori (Led) del ponte in relazione ai vari stati
- **ValveTask**
  si occupa di impostare la posizione delle valvola (Valve) in relazioni ai vari tipi di input selezionati.



Schema d'esempio LampTask

Interfaccia grafica

Per l'interfaccia grafica abbiamo scelto di usare Python e il framework Flask per creare un semplice webserver HTTP, e creare la dashboard come pagina web con HTML, CSS e Javascript.
La scelta di Flask come webserver si è rivelata molto comoda per la gestione delle route http, ma un po' meno comodo per la comunicazione tramite seriale. Durante lo sviluppo del codice abbiamo incontrato delle race-condition di lettura sulla seriale, che abbiamo risolto con una variabile di lock (tipo mutex).
Una alternativa potrebbe essere quella di creare un servizio separato che parli sulla seriale e si interfacci a Flask tramite richieste HTTP, o studiare un sistema per integrare in Flask il servizio di comunicazione seriale su un Thread a se stante.
Il server:

- fornisce la dashboard
- tiene uno storico dei dati del sonar, per il grafico
- risponde alle richieste di aggiornamento della dashboard
- comunica con arduino tramite la porta seriale

La dashboard:

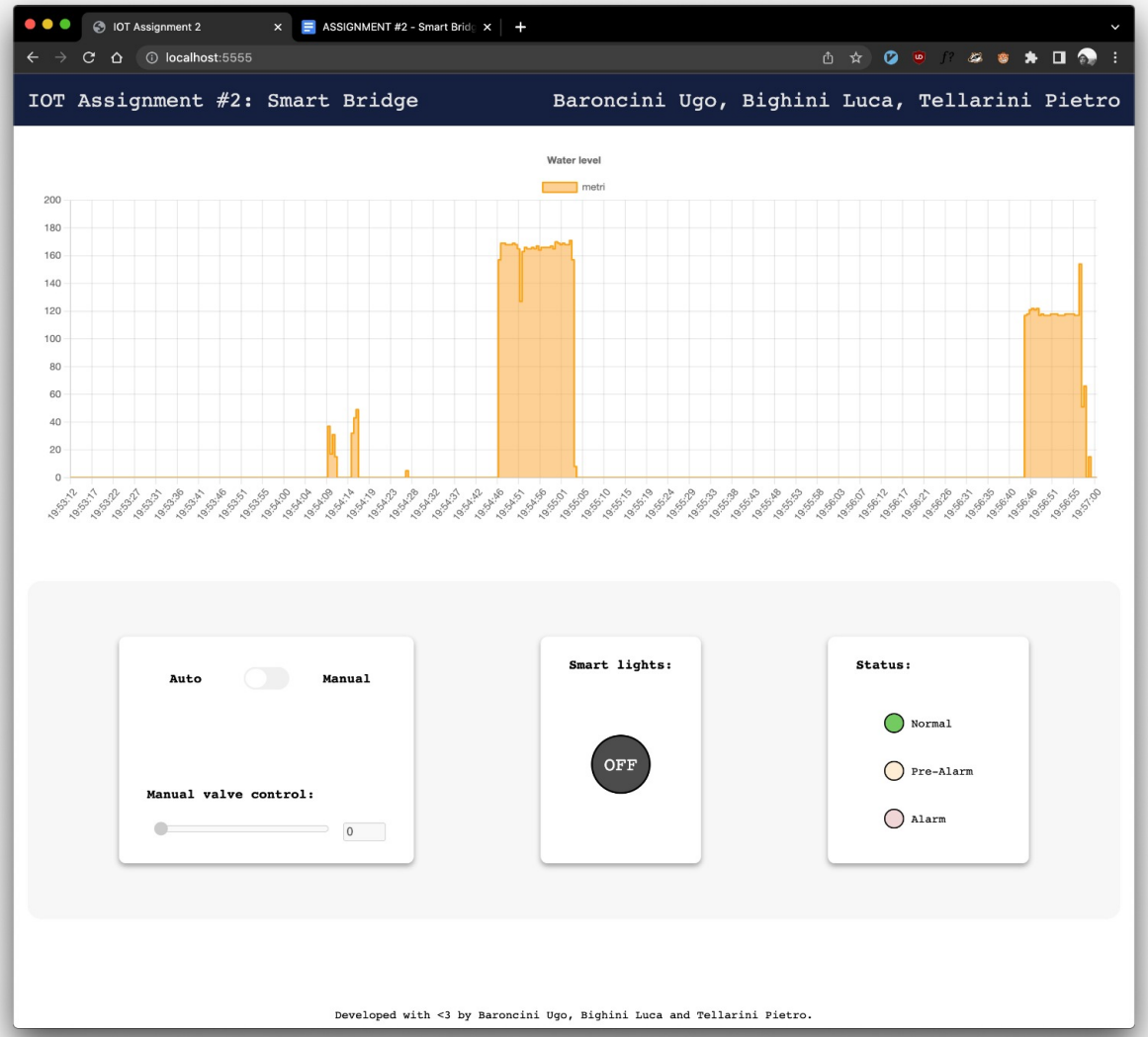
- richiede i dati al server
- mostra correttamente lo stato del sistema e il grafico
- fornisce all'utente la possibilità di pilotare la valvola in stato di allarme.

Per lanciare il server flask:

```
> python -m venv iot-assignment-2-venv
> source iot-assignment-2-venv/bin/activate
> pip install flask pyserial requests
> flask --app bridge-server.py --debug run --port 5555
```

e collegarsi a:

http://localhost:5555

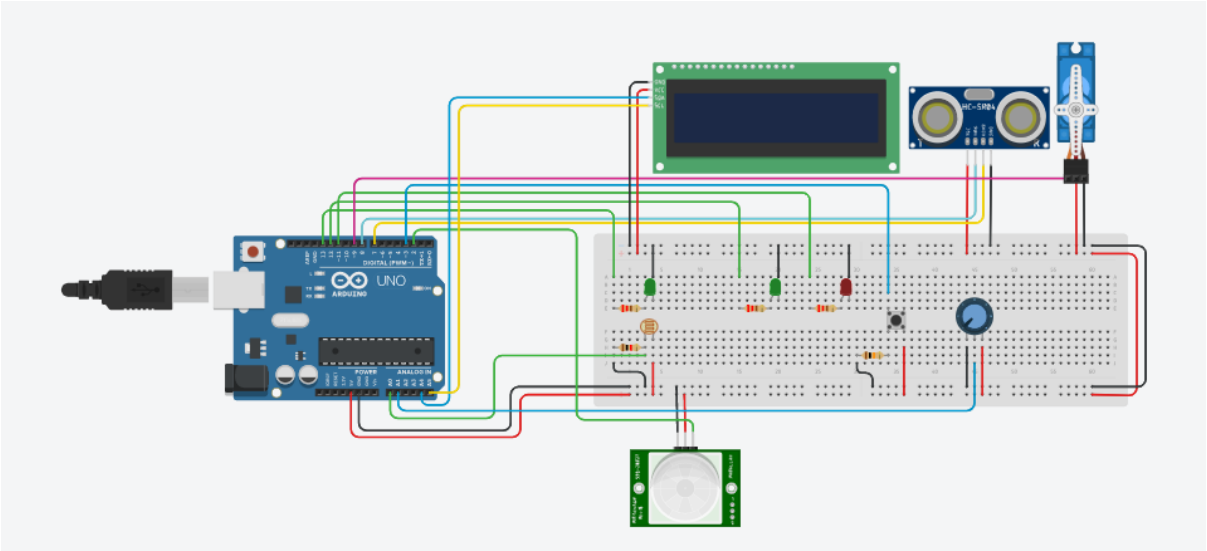

IOT Assignment 2
ASSIGNMENT #2 - Smart Brid
localhost:5555
IOT Assignment #2: Smart Bridge
Baroncini Ugo, Bighini Luca, Tellarini Pietro
Water level
metri
200
180
160
140
120
100
80
60
40
20
0
Auto
Manual
Manual valve control:
0
Smart lights:
OFF
Status:
Normal
Pre-Alarm
Alarm
Developed with <3 by Baroncini Ugo, Bighini Luca and Tellarini Pietro.

Definizioni

Nella cartella Include sono presenti oltre alle definizione dei task, due ulteriori header file:

- Parameters.h
    - lunghezza dei periodi dello scheduler e di ogni task
    - costanti del livello per luce, acqua e tempo
    - costanti per l'inizializzazione dello schermo LCD
- Pins.h
    - dove si trovano i pin relativi ad ogni componente, poi passati come parametri ad ogni Task

Schema Arduino



Video Dimostrativo

Il video è visualizzabile al seguente link YouTube: Link Video.