Diario di lavoro

Luogo	Canobbio
Data	17.02.2020

Lavori svolti

Oggi ho sviluppato la classe che si occuperà di gestire la comunicazione seriale, quindi di aprirla, inviare e ricevere i comandi con relative risposte. La classe che ho sviluppato si chiama **SerialThread** ed ha il metodo principale **run()**, nel quale vengono eseguite le azioni sopra citate.

```
/**
 * Execution of the thread.
 */
@Override
public void run() {

    try {
        this.searchArduino();
        this.openSerialPort();
        this.checkSerialCommunication();

        while (!interrupted()) {
            readCommandWriteResponse();
            writeCommandReadResponse();
        }
    } catch (IOException ioe) {

    }
}
```

La comunicazione avviene finchè la thread non viene interrotta. Per inviare e ricevere i comandi, vengono utilizzati due metodi, **readCommandWriteResponse()** e **writeCommandReadResponse()**.
I comandi vengono inviati tramite una lista, che funziona un po come una coda di comandi, vengono aggiunti alla fine di essa ed ad ogni cilo di comunicazione viene inviato il primo.

Dopo di che ho provato ad integrare questa thread con una servlet. **SerialServlet**, la quale semplicemente ha come attributo (e parametro del costruttore) un oggetto **SerialThread**. Essa ha solamente il metodo HTTP GET implementato, che riceve un parametro **echo** il quale contenuto viene inviato come comando **ECHO** all'Arduino, del quale viene stampato la risposta sulla linea di comando.

```
/**
 * Do at get request.
 *
 * @param req  Http request.
 * @param resp Http response.
 * @throws ServletException Error on servlet.
 * @throws IOException      I/O servlet.
 */
@Override
protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws
ServletException, IOException {

    String string = req.getParameter("echo");
    SerialEchoCommand serialEchoCommand = new SerialEchoCommand(string);

    this.serialThread.addCommand(serialEchoCommand);
```

```
    resp.getOutputStream().println("added");
}
```

La servelet è stata aggiunta ad un webserver di test, il quale avvia la **SerialThread** ed il webserver con la **SerialServlet**.

```
/**
 * Start server.
 *
 * @throws Exception Error while starting server or opening serial port.
 */
public void start() throws Exception {
    startSerialThread();
    Server server = new Server(HTTP_PORT);

    ServletContextHandler context = new
ServletContextHandler(ServletContextHandler.SESSIONS);

    server.setHandler(context);

    context.addServlet(new ServletHolder(new SerialServlet(this.serialThread)), "/");

    server.start();
}
```

Per testare che il webserver funzioni ho deciso di avviarlo e poi eseguire la seguenti richiesta con **curl**.

```
curl http://localhost:8080/\?echo=HelloWorld!
```

La risposta http dovrebbe essere semplicemente

```
added
```

Mentre sul terminale dove è avviato il web server dovrebbe comparire:

```
o
HelloWorld!
```

La **o** significa che la risposta è di tipo OK ed il contenuto è **HelloWorld!** quindi eseattamente ciò che abbiamo inviato nella richeista.

Dopo di che ho implementato e testato i moduli per il microfono ed il telecomando dal lato Arduino. Per testare la parte del telecomando ho creato il seguente programma:

```
while (true) {
    SerialEchoCommand sc = new SerialEchoCommand("HelloWorld!".getBytes());
    sc.write(output);
    SerialCommunication serialCommunication = new SerialCommunication();
    serialCommunication.readUntilEnd(input);
    if (!new String(serialCommunication.getMessage()).equals("null")) {
        System.out.println((char) serialCommunication.getSequence());
        System.out.println(new String(serialCommunication.getMessage()));
    }
    SerialResponse.OK.write(output);
}
```

Per testare che funzioni bisogna semplicemente collegare il pin 5 con il pin 3 tramite una restistenza e poi il pin 5 con GND. L'output del programma dovrebbe essere:

```
r
1
```

```
r
0
```

Problemi riscontrati e soluzioni adottate
-

Punto della situazione rispetto alla pianificazione
Sono in linea con la pianificazzione, ma sono un po indietro con la documentazione. Attivita 12.

Programma di massima per la prossima giornata di lavoro
Implementare la parte di Arduino per la gestione delle frequenze.