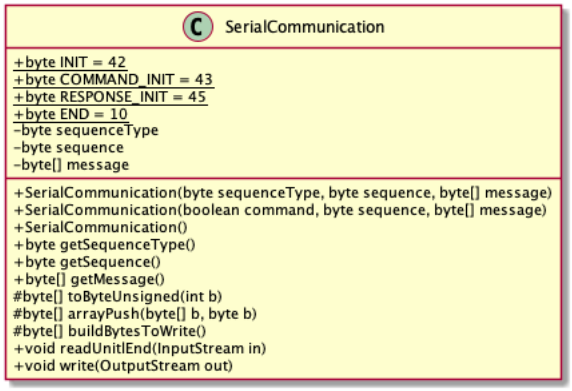
Diario di lavoro

Luogo	Canobbio
Data	12.02.2020

Lavori svolti

Oggi ho iniziato con lo sviluppo vero e proprio delle classi Java del protocollo.
Le quali sono implementate in maniera astratta, cosí da facilitare il reciclo del codice. Per questo motivo ho iniziato scrivendo una classe che mi permetta di inviare e ricevere i messaggi tramite degli Input/Output stream, mentre scrivevo la classe mi è venuto in mente che bisognerebbe distinguere l'invio di una sequenza di comando da una di risposta, quindi ho eseguito una piccola modifica del protocollo, che poi andra fatta anche lato Arduino. Per definire se una sequenza è un comando oppure una risposta bisogna inviare prima di esso il byte 43 per i comandi ed il byte 45 per le risposte.

Il diagramma UML della classe sviluppata:



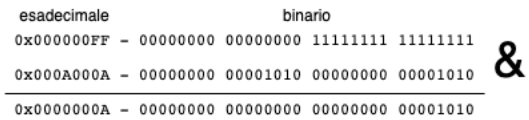
Questa classe permette di inviare e ricevere messaggi secondo il protocollo sviluppato, tramite degli input/output stream di qualunque tipo.

Una parte molto importante della classe sarebbe la parte relativa, alla conversione dei dati ricevuti tramite input stream da int a byte, che devono essere unsigned. Quindi ho creato il seguente metodo per questa conversione:

```
/**
 * Bytes for unsigned byte casting.
 */
public static final int TO_BYTE = 0x000000FF;

/**
 * Cast int to byte unsigned.
 *
 * @param b Byte to parse.
 * @return Unsigned byte of int.
 */
protected byte toByteUnsigned(int b) {
    return (byte) (TO_BYTE & b);
}
```

In questo metodo viene fatta un'operazione AND (&) bit – bit, che significa che vengono comparati i singoli bit e vengono lasciati ad uno quelli che in entrambe le parti sono a 1.



Dopo aver fatto questa comparazione si ha il byte ricevuto tramite **InputStream** Anche nel caso in cui il byte ha il primo byte settato ad **1**, altrimenti verrebbe sballato tutto quanto.

Per testare questa classe ho deciso di utilizzare il sistema del Unit Testing (**Junit**), siccome testare tramite arduino sarebbe più complesso, siccome dovrei scrivere dei programmi fatti apposta per eseguire i test, con un display LCD, siccome la porta seriale sarebbe gia utilizzata.
Per utilizzare Junit con gli input/output stream ho trovato una guida su **https://blog.ostermiller.org/convert-a-java-outputstream-to-an-inputstream/**, di queste soluzioni ho deciso di adottare la prima. Siccome mi sembra la più facile e veloce da implementare.

Test del metodo **write()** della classe **SerialCommunication**:

```
@Test
public void testWrite() throws IOException {
    ByteArrayOutputStream out = new ByteArrayOutputStream();

    SerialCommunication sc = getSerialCommunicationInstance();
    sc.write(out);
    byte[] bytes = out.toByteArray();

    assertArrayEquals(sc.buildBytesToWrite(), bytes);
}
```

Problemi riscontrati e soluzioni adottate

Testare gli Input/Output Stream senza la seriale, quindi ho utilizzato le unit testing.
(**https://blog.ostermiller.org/convert-a-java-outputstream-to-an-inputstream/**)

Punto della situazione rispetto alla pianificazione

Sono giusto rispetto alla pianificazione. (attivita 12)

Programma di massima per la prossima giornata di lavoro

Andare avanti con lo sviluppo della comunicazione seriale.