Diario di lavoro

Luogo	Canobbio
Data	10.02.2020

Lavori svolti

Oggi ho continuato lo sviluppo del programma di test, della comunicazione seriale.
In maniera che l'arduino ritorni sulla porta seriale quello che riceve. Come un echo server.

Codice arduino:

```
// setup arduino
void setup() {
  // open channel at 9600 bit/s
  Serial.begin(9600);
  // send communication start char
  Serial.write(42);
}

// loop of the arduino
void loop() {
  // Read from serial
  String s = Serial.readStringUntil('\n');
  // wirte on serial each character readed before
  for (int i = 0; i < s.length(); i++) {
    Serial.write(s[i]);
  }
  // write return character
  Serial.write('\n');
}
```

Codice java:

```
public class App {
    public static final byte INIT = 42;

    public static final byte RETURN = (byte) '\n';

    public static void main(String[] args) {
        // search arduino on serial
        SerialPort serialPort = null;
        for (SerialPort serial : SerialPort.getCommPorts()) {
            if (serial.getDescriptivePortName().contains("Arduino")) {
                serialPort = serial;
            }
        }

        // exit if arduino not found
        if (serialPort == null) {
            System.out.println("Arduino not found");
            return;
        }

        // set new higher timeouts
        serialPort.setComPortTimeouts(SerialPort.TIMEOUT_READ_BLOCKING, 100000000,
10000000);

        // open serial port communication
        if (!serialPort.openPort()) {
```

```
            System.out.println("Error while opening communication");
        }

        // set streams (I/O)
        InputStream input = serialPort.getInputStream();
        OutputStream output = serialPort.getOutputStream();

        int read;
        try {
            // wait until INIT char
            while ((read = input.read()) != INIT) { }

            // wirte HelloWorld!\n on serial
            output.write("HelloWorld!\n".getBytes());

            Thread.sleep(100);

            // read and print on CLI all chars until return
            while ((read = input.read()) != RETURN) {
                byte c = (byte)(0x000000FF & read);
                System.out.print((char)c);
            }

            // close streams
            output.close();
            input.close();
        } catch (IOException ioe) {
            ioe.printStackTrace();
            System.out.println(ioe.getMessage());
        } catch (InterruptedException ie) {

        }
    }
}
```

Dopo di che ho iniziato il design dei componenti che dovranno essere aggiunti all'interno della scatola. I quali sono i seguenti:

- Raspberry PI
- Microfono
- Alimentatore Raspberry PI
- Alimentatore interno

I quali potrebbero venire posizionati circa come nell'immagine sottostante.

Per quanto riguarda i componenti il wireless, verrà emesso dal Raspberry PI, il quale quindi dovrà emettere un segnale wireless, con un server DHCP e DNS, anche sull'interfaccia Ethernet, cosi che sia possibile utilzzarlo anche via cavo.
Per il servizio DHCP utilizzero **dhcpd**, siccome lo conosco già. Mentre per quanto riguarda il servizio DNS utilizzero **bind named** che anche in questo gia conosco già. La parte dei servizi verrà fatta alla fine, siccome ho bisogno di collegare ancora il raspberry alla rete delle scuola, quindi non posso avviare il servizio DHCP.

Siccome ho finito la parte di progettazione comincio gia con l'implementazione del progetto, quindi passo all'attività 12 del progetto.
Ho iniziato dal lato arduino della trasmissione.

Quindi ho creato il file **acc_serial.ino** che continene i metodi relativi alla comunicazione seriale dell'arduino.

```
#define SERIAL_SPEED 9600                    // serial speed of communication (baud)
#define SERIAL_INIT 42                       // serial initialized char
#define SERIAL_END_OF_COMMAND '\n'           // End of the command char

/**
 * Initialization of serial communication.
 */
void serialSetup() {
  // open serial channel
  Serial.begin(SERIAL_SPEED);
  // send communication start char
  Serial.write(SERIAL_INIT);
}

/**
 * Read command from serial.
 */
void readCommand() {
  // read command from serial line
  String s = Serial.readStringUntil(SERIAL_END_OF_COMMAND);

  // if command exists
```

```
  if (s.length() > 0) {
    // set command char to command variable
    command = s[0];
    // set command value to command value variable
    commandValue = s.substring(1);
  }
}

/**
 * Write command to serial.
 *
 * @param command command char to write to serial
 * @param value value of the command to wirte to serial
 */
void writeCommand(char command, String value) {
  // write command to serial
  Serial.write(command);

  // write all chars of the value to serial
  for (int i = 0; i < value.length(); i++) {
    Serial.write(value[i]);
  }

  // write end of command char to serial
  Serial.write(SERIAL_END_OF_COMMAND);
}
```

Dopo di che ho testato il codice che ho scritto, creando un piccolo programma da inserire sull'arduino, che come nell'esempio soprastante fa da echo server. Cosi da poter utilizzare il codice java già presente. Il quale si presenta come il sottostante:

```
char command;
String commandValue;

// setup arduino
void setup() {
  serialSetup();
}

// loop of the arduino
void loop() {
  readCommand();
  writeCommand(command, commandValue);
}
```

Eseguendo il codice sull'arduino e sul computer la parte java. Il programma funzionava correttamente, però ho deciso di rinominare i metodi , **readCommand() → serialReadCommand()** e **writeCommand() →serialWriteCommand()**, questo per evitare che si possa fare confuzione fra i vari metodi che verrano creati in futuro.
Poi ho inziato a sviluppare la parte di analisi dei comandi dell'arduino. Quindi creare il comando di base echo, che verrà utilizzato per i test di connessione.

Problemi riscontrati e soluzioni adottate

-

Punto della situazione rispetto alla pianificazione

Sono leggermente avanti con la pianificazione, sono riuscito a finire oggi la parte di progettazione ed iniziare leggermente quella di implementazione. Attività 12.

Programma di massima per la prossima giornata di lavoro
Continuare con lo sviluppo della connessione fra Arduino e Java. Comunicazione di base dei comandi con Java. Attivita 12