Documentazione Software Turbidimetro

Luca Chiocchetti
Mat. 564645

Indice

1 Introduzione

Il presente progetto si propone di fornire un'esaustiva trattazione sul turbidimetro, uno strumento di rilevamento della torbidità che riveste un ruolo di primaria importanza in svariate discipline scientifiche e applicazioni industriali. In un contesto in cui la qualità dell'acqua e la ricerca scientifica rappresentano nodi cruciali, il turbidimetro emerge come un dispositivo fondamentale per l'analisi e il monitoraggio dei sedimenti presenti nell'acqua.

Il turbidimetro può avere una delle due configurazioni:

- Fisso: non si posta dalla sua sede.
- Mobile: si può spostare e la sua posizione è rilevata tramite GPS.

1.1 Presentazione del sistema

Il sistema si compone di due parti:

- Turbidimetro: Sistema di rilevamento della torbità dell'acqua composto da:
  - Rasbperry pi zero WH.
  - Wittypi 4 mini per la gestione del consumo energetico.
  - Dispositivo LTE SIM7600X con rilevamento GPS, previsto solo nella versione mobile del turbidimetro.
  - Alimentazione: fissa o powerbank.
- Server: Sito costruito per la visualizzazione dei dati raccolti inviati dal turbidimentro.

1.2 File del turbidimetro

Vediamo nel dettaglio i file che compongono il software di gestione del turbidimetro.

1.2.1 config.ini

File di configurazione del turbidimetro
In fase di configurazione i campi vanno inizializzati con gli opportuni valori con cui si intende caratterizzare il turbidimetro e con le informazioni per la connessione al server che si ha a disposizione
Si compone di 3 parti:

- General: contiene le informazioni generali del Raspberry Pi
- Turbidimeter: contiene le informazioni sul comportamento del turbidimetro
- Server: contiene i dettagli di connessione con il server del'organizzazione

General

```
[general]
 user=your username
```

user contiene il nome utente del raspberry pi, tale nome viene scelto dall'utente nel momento dell'installazione del sistema operativo raspbian sul raspberry pi.
Vedere la sezione 2 per maggiori dettagli.

Turbidimeter

```
[turbidimeter]
turbidimeterID=1
```

Contiene l'ID assegnato al turbidimetro.
Offre la possibilità di avere più turbidimetri nella propria organizzazione variando l'ID assegnato ai vari turbidimetri

```
dataReadInterval=30
```

Valore dell'intervallo di tempo espresso in secondi tra le letture dei sensori, della tensione e se mobile=true anche della posizione, è significativo solo se numberOfsampling è maggiore di 1.

```
numberOfSampling=1
```

numero di prelievi effettuati in un unica accensione del turbidimetro.
Offre la possibilità di effettuare più prelievi una volta che il turbidimetro si sveglia, tra un prelievo e l'altro passano dataReadInterval secondi.
ATTENZIONE: La durata in secondi data da

$$dataReadInterval \times numberOfSampling$$

deve essere inferiore di 120 secondi del tempo in cui il turbidimetro rimane acceso, questo per garantire la correttezza dei risultati.

```
mobile=True
```

Indica se il turbidimetro è nella sua configurazione mobile se il valore dato è True o in quella fissa se il valore è False.
Si raccomanda di scrivere True e False con la maiuscole iniziali.

```
wait=60
```

valido solo se mobile=true, è il tempo di attesa prima di prelevare la posizione GPS. Tale attesa serve per permettere al GPS di collegarsi a un satellite per ottenere la posizione.

Server

```
[server]
ipAddr=your ip address
```

Il campo va inizializzato con l'indirizzo IP del server che si ha a disposizione.

```
userServer=your username server
```

nome utente con cui effettuare l'accesso al sito web o alla macchina virtuale. La password per la connessione verrà in un secondo momento.

```
siteName=your site name
```

Da inizializzare con il nome del sito web dove vengono visualizzati i dati legati al turbidimetro.

1.2.2 welcomeTurbi.sh

Lo script ha il compito di installare nei percorsi corretti, le librerie per la gestione dei sensori e delle periferiche.
Inoltre, si occupa di chiedere la password del server o della macchina virtuale usando lo script 1.2.6

1.2.3 connectLTE.sh

Lo script effettua la connessione LTE utilizzando la scheda LTE SIM7600X, è necessaria una sim 4G e con un piano tariffario con GB di internet.

1.2.4 sendATCommands.py

Utilizza l'interfaccia seriale per mandare un comando AT alla scheda LTE SIM7600X, questo programma viene utilizzato da connectLTE.sh per creare la connessione.

1.2.5 main.py

Tale programma è il cuore del software del turbidimetro, infatti, utilizzando le librerie installate con 1.2.2 effettua:

- Prelievo del valore dei sensori TSL25911FN con led spento

```
      led.off()

sensor1.Lux
sensor3.Lux
sensor1.TSL2591_SET_LuxInterrupt(50, 200)
sensor3.TSL2591_SET_LuxInterrupt(50, 200)

infraredOFF1 = sensor1.Read_Infrared
infraredOFF3 = sensor3.Read_Infrared

visibleOFF1 = sensor1.Read_Visible
visibleOFF3 = sensor3.Read_Visible

fullSpectrumOFF1 = sensor1.Read_FullSpectrum
fullSpectrumOFF3 = sensor3.Read_FullSpectrum
```

- Prelievo dei valori dei sensori TSL25911FN con led acceso. La sleep serve a garantire il tempo necessario affinché il led sia correttamente acceso

```
      led.on()
time.sleep(5)

sensor1.Lux
sensor3.Lux
sensor1.TSL2591_SET_LuxInterrupt(50, 200)
sensor3.TSL2591_SET_LuxInterrupt(50, 200)

infraredON1 = sensor1.Read_Infrared
infraredON3 = sensor3.Read_Infrared

visibleON1 = sensor1.Read_Visible
visibleON3 = sensor3.Read_Visible
```

```
fullSpectrumON1 = sensor1.Read_FullSpectrum
fullSpectrumON3 = sensor3.Read_FullSpectrum
```

- prelievo della tensione di alimentazione del raspberry pi. Tale tensione viene scritta sul file "voltage.txt" dallo script "afterStartup.sh", viene quindi aperto il file, letta la tensione, salvata dentro a una variabile e poi scritta su file txt del prelievo insieme agli altri valori.

```
  file = open("voltage.txt", "r")
voltage = file.readline()
file.close()
```

- prelievo della posizione GPS se $mobile = True$

```
if mobile == "True":
print("prelevo la posizione del dispositivo")
if __name__=="__main__":

        ser = serial.Serial("/dev/ttyUSB2", 115200)

        argument=sys.argv[1]


        try:

                position = send_command(argument)


        except:
                print("Request failed")
                ser.close()
                serial.Serial("/dev/ttyUSB2", 115200)
```

1.2.6 makePasswordRsync

Lo script chiede all'utente di inserire la password per il collegamento con la macchina virtuale o con il sito web.
La password viene crittografata usando l'AES a 256 bit e salvata su file per poi essere utilizzata dalla Rsync nello script 1.2.7.
Lo script non fa eco della password inserita.

1.2.7 sync.sh

Utilizza le informazioni fornite dal file di configurazione 1.2.1 e dallo script 1.2.6 per sincronizzare una cartella locale con una cartella in remoto.
La sincronizzazione avviene inviando solo le differenze tra la cartella locale e quella remota, tale metodo permettere di non rendere troppo pesante l'operazione.
Nella nostra configurazione le cartelle sincronizzate hanno lo stesso nome e si chiamano "values" ma è possibile cambiarle il nome a patto di cambiare i percorsi.

- Percorso locale:

```
"/home/"$user"/Desktop/progetto/values/"$turbidimeterID
```

- Percorso remoto:
  Da cambiare in base a ciò che si utilizza come hosting del sito web.

```
$userServer@$ipAddr:/opt/lampp/htdocs/$siteName/values
```

1.2.8 startTurbi.sh

Lo script viene lanciato da 1.2.9 e manda in esecuzione i file visti in precedenza per effettuare le attività del turbidimetro, utilizza le informazioni riportate in 1.2.1.
In dettaglio svolge i seguenti comandi:

- Testa la condizione su mobile per capire in che configurazione si trova il turbidimetro.

```
if test $mobile = "True"
  then
```

- Se la condizione è verificata allora svolge le seguenti istruzioni:

```
echo "mobile turdimeter"
```

  - Accende il GPS

```
python3 sendATCommands.py at+cgps=1
```

- Attende che il GPS si colleghi a un satellite per rilevare la propria posizione

```
sleep $wait
```

- Svolge il programma "main.py" dandogli come argomento il comando AT per la richiesta di informazioni sulla posizione

```
python3 main.py at+cgpsinfo
```

- Spegne il GPS

```
python3 sendATCommands.py at+cgps=0
```

- Manda il esecuzione lo script "connectLTE.sh" per connettere il turbidimetro alla rete cellulare

```
sudo ./connectLTE.sh
```

- Se la condizione è falsa manda solamente in esecuzione il programma "main.py"

```
 else
  echo "Fixed turbidimeter"
  python3 main.py
fi
```

- Cancella il file "voltage.txt" in modo tale da evitare che una successiva scrittura possa compromettere il formato del file.
  Infatti, vogliamo essere certi che la tensione letta durante l'esecuzione di "main.py" sia quella attuale del turbidimetro.

```
rm voltage.txt
```

- Manda in esecuzione lo script "Sync.sh" in modo tale da sincronizzare la cartella locale con la cartella in remoto inviando i dati appena prelevati.

```
 sudo ./sync.sh
echo "#--------End of turbidimeter activities-------#"
echo "#------the turbidimeter will soon shut down------#"
```

1.2.9 afterStartup.sh

Viene copiato nel file afterStartup della libreria del wittyPi 4 mini.
Contiene le istruzioni che il turbidimetro effettua ad ogni suo risveglio:

- Legge la tensione sul Raspberry pi utilizzando le librerie del Witty Pi 4 mini e la scrive sul file voltage.txt

```
   dir=/home/$user/wittypi
. $dir/utilities.sh
. $dir/gpio-util.sh
echo $(get_output_voltage) > /home/$user/Desktop/progetto/voltage.txt
```

- Manda in esecuzione "startTurbi.sh"

```
cd /home/$user/Desktop/progetto/
sudo ./startTurbi.sh
```

1.2.10 Salvataggio su file e il loro formato

Nella cartella "values" è possibile trovare altre sottocartelle ordinate per ID del turbidimetro, ogni cartella ID ha al suo interno le cartelle con le date dei prelievi e ogni cartella del giorno ha al suo interno i file nominati con il timestamp del prelievo. Escludendo le cartelle con l'ID, le altre cartelle vengono create dal programma "main.py"

```
if not os.path.exists("values"):
   os.makedirs("values")

if not os.path.exists(os.path.join("values", turbidimeterID)):
 os.makedirs(os.path.join("values", turbidimeterID))
 current_date = time.strftime("%Y-%m-%d")
if not os.path.exists(os.path.join(os.path.join("values", turbidimeterID),
 current_date)):
```

```
   os.makedirs(os.path.join(os.path.join("values", turbidimeterID),
   current_date))

 current_time = time.strftime("%H-%M-%S")
 filename = "values/" + turbidimeterID + "/" + current_date + "/" +
 "val" + current_time + ".txt"
```

Successivamente nel file vengono salvate tutte le informazioni raccolte:

```
file = open(filename, "w")
file.write(str(infraredOFF1) + ' ' + str(visibleOFF1) + ' '
+ str(fullSpectrumOFF1) + ' ' + str(infraredON1) + ' ' + str(visibleON1) +
 ' ' + str(fullSpectrumON1))
file.write(str(infraredOFF3) + ' ' + str(visibleOFF3) + ' '
+ str(fullSpectrumOFF3) + ' ' + str(infraredON3) + ' ' + str(visibleON3) +
 ' ' + str(fullSpectrumON3) + ' ')
file.write(str(voltage))

if mobile == "True":
 div = position.split(":")
if len(div) == 1:
 print("posizione non prelevata, il gps non riesce a trovare la posizione\n")
else:
 info = div[1].split(",")
 latitude = [float(info[0]), info[1]]
 longitude = [float(info[2]), info[3]]
 file.write(str(latitude[0])  + ' ' + str(longitude[0]))
```

Tali informazioni variano in base al valore di mobile Il formato di un file.txt è il seguente:

- Nel prima riga abbiamo le informazioni relative ai sensori: letture fatte con led acceso e spento e alla tensione sul raspberry pi, quindi alle informazioni presenti in entrambe le configurazioni del turbidimetro.

  ```
  88 1703998 1704024 89 1703999 1704025814 14942794
  14943022 814 15073864 15074094 5
  ```

- Nella seconda riga abbiamo le informazioni riguardanti la posizione del turbidimetro, relative solo alla configurazione mobile.

  ```
  4343.271562 1023.42445
  ```