The Internet of Things: Lab Activities

Course website: http://site.unibo.it/iot

Prof. Luciano Bononi
luciano.bononi@unibo.it

Prof. Marco Di Felice
marco.difelice3@unibo.it

MASTER DEGREE IN COMPUTER SCIENCE
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING, UNIVERSITY OF BOLOGNA, ITALY

Argomento 1

Programmazione Sistemi Embedded mediante board di prototipazione (Arduino)

Preliminari

1. Connettersi all'URL : https://www.tinkercad.com
2. Cliccare su Students → Join your class
3. Inserire il **codice** della classe fornito dal docente
4. Inserire il **nickname** dello studente fornito dal docente

Esercizio 1

Scrivere un **programma** (sketch) che:

- Stampa una scritta ogni **Y** ms:
  «**Hello world , current delay interval: #Y**»
  Il valore iniziale di Y è pari a 5000 millisecondi
- Ad ogni iterazione si modifica il valore del Y nel modo seguente:
  - ❖ Se Y è inferiore a 100 ms, allora Y viene raddoppiato (Y=Y*2)
  - ❖ Se Y è maggiore di 5000 ms, allora Y viene dimezzato (Y=Y/2)

Esercizio 1 -- HELP

```
setup() {
   Serial.begin(115200); // Inizializza la porta seriale
}

begin {
   Serial.println(«My string»); // Stampa su seriale
   Serial.print(«My string»); // Stampa su seriale
   delay(1000); // Setta lo sleep time a 1000 secondi
}
```

Esercizio 2

Progettare uno schema di sistema **embedded** (Hw/Sw) composto da:

- Arduino UNO
- Sensore di prossimità (PIR)

Lo sketch di Arduino deve (continuamente):

- Acquisire i dati dal PIR
- Stampare a video <<Motion Detected>> ogni qualvolta si rileva un movimento dal sensore

Esercizio 2 -- HELP

```
setup() {
    pinMode(#NUMERO_PIN, INPUT | OUTPUT);
   //Attiva un PIN con ID #NUMERO_PIN in lettura o scrittura
}
begin {
    mia_variabile=digitalRead(#NUMERO_PIN);
    //Legge da PIN #NUMERO_PIN un valore digitale (HIGH | LOW)
    digitalWrite(#NUMERO_PIN, HIGH | LOW);
   //Scrive su PIN #NUMERO_PIN un valore digitale (HIGH | LOW)
}
```

Esercizio 3 (1/2)

Progettare uno schema di sistema **embedded** (Hw/Sw) composto da:

- Arduino UNO
- Sensore di prossimità (PIR)
- LED

Esercizio 3 (2/2)

Lo sketch Arduino deve (continuamente):

1. Acquisire i dati dal PIR
2. Stampare a video <<Motion Detected>> ogni qualvolta si rileva un movimento dal sensore
3. Accendere il LED in caso di cambio di stato: no movimento → movimento
4. Spegnere il LED in caso di cambio di stato: movimento → no movimento

Argomento 2

Acquisizione dati mediante protocolli di messaging IoT MQTT/CoAP

Esercizio 4 (1/2)

Scrivere uno **script di acquisizione dati** da un sensore di temperatura/umidità mappato sul protocollo MQTT

```
Indirizzo IP broker: 130.136.2.70
topic_1: sensor/temperature
topic_2: sensor/humidity
```

Fare in modo che lo script:

- Acquisisca dati in continuo
- Ogni X misurazioni acquisite (sia per la temperatura sia per l'umidità):
  - ✓ Calcoli il valore minimo, massimo e medio
  - ✓ Resetti i contatori

Esercizio 4 (2/2)

Implementazione con JS/Node

https://www.npmjs.com/package/mqtt

```
npm install mqtt
```

```
var client  = mqtt.connect(MQTT_ADDR,{clientId: 'clientJS', protocolId:
           'MQIsdp', protocolVersion: 3, connectTimeout:1000, debug:true});
```

Implementazione con Python

https://pypi.org/project/paho-mqtt/

```
pip install paho-mqtt
```

Esercizio 5 (1/2)

Scrivere uno **script di acquisizione dati** da un sensore di luminosità mappato sul protocollo CoAP

`Indirizzo IP CoAP server: coap://130.136.2.70/light`

Fare in modo che lo script:

- Interroghi il server CoAP ogni 3 secondi
- Ogni X misurazioni acquisite:
  - ✓ Calcoli il valore minimo, massimo e medio
  - ✓ Resetti i contatori

Esercizio 4 (2/2)

Implementazione con JS/Node

https://www.npmjs.com/package/coap

```
npm install coap
```

Implementazione con Python

https://github.com/Tanganelli/CoAPthon

```
pip install CoAPthon
```

Argomento 3

Acquisizione dati mediante approcci Web of Things

Esercizio 5 (1/2)

Scrivere uno **script di acquisizione dati** che recupera l'ultimo dato del valore di temperatura dal seguente canale ThingSpeak:

ID Canale: **1039278**
ReadAPIKey: **A5NNBPICIYI09O9G**
Field1

https://thingspeak.com

Esercizio 5 (2/2)

Implementazione con JS/Node

https://www.npmjs.com/package/thingspeakclient

```
npm install thingspeakclient
```

Implementazione con Python

https://pypi.org/project/thingspeak/

```
pip install thingspeak
```

Esercizio 5 (2/2)

Scrivere uno **script di acquisizione dati** che, nel caso in cui l'ultimo dato del valore di temperatura sia più alto di una soglia prefissata, fa emettere un tone personalizzato da un **Passive Buzzer** connesso al sistema ThingSpeak

ID Canale: **1039278**
WriteAPIKey: **CXCTYRARAZSC58QU**
Field3

Formato di ogni comando (entry da scrivere):

Name; TONE1; … TONE2; … TONEM (M<5 valori)

Marco; 45; 67;

LISTA DEI «TONI»: https://github.com/lbernstone/Tone/blob/master/src/pitches.h

Argomento 4

Acquisizione dati mediante approcci W3C Web of Things

Esercizio 6 (1/3)

Scrivere un'application che recupera la Thing Description della Thing Sensor e si sottoscrive agli eventi relativi al dato di temperatura:

Indirizzo della Thing Sensor

`http://iot2020.cs.unibo.it:8081`

Protocollo: HTTP

Esercizio 6 (2/3)

Scrivere un'application che recupera la Thing Description della Thing Buzzer e invoca l'action per attivarlo:

Indirizzo della Thing Buzzer (Protocollo: CoAP)
`http://iot2020.cs.unibo.it:8080`

- Formato di ogni comando (entry da scrivere):
  `VostroName; TONE1; … TONE2;  … TONEM (M<5 valori)`
  `Marco; 45; 67;` → LISTA DEI «TONI»: https://github.com/lbernstone/Tone/blob/master/src/pitches.h
- Formato del **payload** della action (JSON):
  {«value»: CMD_STRING }, dove CMD_STRING è il formato di cui sopra
  {«value»: «`Marco; 45; 67;`» }

Esercizio 6 (3/3)

Scrivere una mash application che recupera la Thing Description sia della Thing Buzzer sia della Thing Sensor e attiva il Buzzer solo se il dato di temperatura è maggiore di una certa soglia prefissata.

Indirizzo della Thing Buzzer (COAP)

`http://iot2020.cs.unibo.it:8080`

Indirizzo della Thing Sensor (HTTP)

`http://iot2020.cs.unibo.it:8081`

Argomento 5

Gestione di serie temporali mediante Time-series Database

Esercizio 7 (1/2)

Costruire una dashboard in **Grafana** in modo da visualizzare i dati del sensori di umidità e temperatura, memorizzati su un apposito DB INFLUX. Nello specifico, costruire una **dashboard** con due grafici della time-series (uno per ogni sensore). Ogni grafico deve avere due linee:

- ✓ Una relativa alle misurazioni attuali
- ✓ Una relativa alla media, calcolata su una finestra dell'ultimo minuto di misurazione

Gestire il **refresh** automatico della dashboard ogni 2 secondi.

Esercizio 7 (2/2)

Costruire una dashboard in **Grafana** in modo da visualizzare i dati del sensore di umidità, memorizzati su un apposito DB INFLUXDB.

Grafana: http://iot2020.cs.unibo.it:8500
Bucket: iotdemo
Measurement: humSensor
Measurement: tempSensor

Esercizio 8 (1/2)

Scrivere uno **script di processamento dati IOT che:**

1) Acquisca dati da un sensore di temperatura/umidità mappato sul protocollo MQTT

```
Indirizzo IP broker: 130.136.2.70
topic_1: sensor/temperature
topic_2: sensor/humidity
```

2) Memorizzi i dati acquisiti su time-series DB INFLUX:

```
Indirizzo IP INFLUX: 130.136.2.70:8999
org: iotclass       db: iotdemo
measurement: tempValue (per valori di temperatura)
measurement: humValue (per valori di umidità)
```

Aggiungere un opportuno tag per idenficare i propri dati: (es. user=XXX)

Esercizio 8 (2/2)

Implementazione con JS/Node

```
https://github.com/influxdata/influxdb-client-js
npm install --save @influxdata/influxdb-client
```

Implementazione con Python

```
https://github.com/influxdata/influxdb-client-python
pip install influxdb-client
```

Argomento 5

Processamento ed Analisi di serie temporali

Esercizio 9 (1/4)

Scrivere uno **script di analisi dati IOT che:**

1) Popola un array di 1000 valori, nella maniera seguente:
   - ✓ Quando il valore della serie diventa minore di 13, la serie entra in modalità INCREASE
   - ✓ Quando il valore della serie diventa maggiore di 30, la serie entra in modalità DECREASE
   - ✓ In modalità INCREASE: serie(t+1)=serie(t)+0.1
   - ✓ In modalità DECREASE: serie(t+1)=serie(t)-0.1
   - ✓ Valore iniziale: serie(0)=13

Esercizio 9 (2/4)

2) Predice il valore degli ultimi 5 valori della serie attraverso il metodo ARMA(1,0,1)

3) Calcola il valore del **Mean Square Error** (MSE), confrontando i valori reali della serie con quelli predetti.

4) Ripete l'analisi applicando il metodo ARIMA(1,1,1)

Esercizio 9 (3/4)

POPOLAMENTO ARRAY

```
const MAX_TEMP=30;
const MIN_TEMP=13;
const RATE=0.1;
const NUM_POINTS_TO_FORECAST=5;
const ARRAY_SIZE=1000;
var temp=MIN_TEMP;
var modeIncrease=true;
const ts = Array(ARRAY_SIZE).fill(0).map((v, i) => {
        if (modeIncrease)
                temp+=RATE;
        else
                temp-=RATE;
        if ((temp>=MAX_TEMP && modeIncrease))
                modeIncrease=false;
        else if (temp<=MIN_TEMP && !modeIncrease)
                modeIncrease=true;


        return temp;
});
```

Esercizio 9 (4/4)

Implementazione con JS/Node

https://www.npmjs.com/package/arima

```
npm install arima
```

Implementazione con Python

https://www.statsmodels.org/stable/generated/statsmodels.tsa.arima_model.ARIMA.html

Argomento 6

Esercitazione riassuntiva

Esercizio 10 (1/4)

Scrivere uno **script di analisi dati IOT che:**

1) Acquisisce dati dal canale MQTT seguente:

- ✓ Host: iot2020.cs.unibo.it
- ✓ Port: 1883
- ✓ Username: iot2020
- ✓ Topic: arduino

Esercizio 10 (1/5)

Scrivere uno **script di analisi dati IOT che:**

2) Scrive i dati acquisiti sul DB Influx:

- ✓ Host: iot2020.cs.unibo.it
- ✓ Port: 8999
- ✓ Bucket name: iotex
- ✓ Per la scrittura di misurazioni sul DB:
  - ✓ Usare l'API fornita dal file persister_influx.db
  - ✓ Effettuare una HTTP GET a questo servizio:
    - ❖ http://iot2020.cs.unibo.it:8005
    - ❖ Parametri: username, sensorname, value

Esercizio 10 (2/5)

Scrivere uno **script di analisi dati IOT che:**

2) Scrive i dati acquisiti sul DB Influx:

- ✓ Per l'accelerometro e del magnetometro, si chiede di salvare sul DB il valore aggregato della magnitude, definito come:

  - ❖ $\text{Magnitude(acc)}=\text{Sqrt}(acc_x^2 + acc_y^2 + acc_z^2)$

Esercizio 10 (3/5)

Scrivere uno **script di analisi dati IOT che:**

3) Effettua le predizione (forecast) del prossimo valore del barometro, della temperatura e dell'umidità, sulla base degli ultimi 30 valori ricevuti.

- Utilizzare il metodo ARIMA, con il file arima.js messo a disposizione

Esercizio 10 (4/5)

Scrivere uno **script di analisi dati IOT che:**

4) Costruisce apposita dashboard in GRAFANA, con un grafico per ciascun sensore meteo (temperatura, umidità, pressione).

- All'interno di ciascun grafico, deve essere mostrato:
- Valore attuale del sensore
- Valore della predizione, calcolata al punto 3

Esercizio 10 (5/5)

Scrivere uno **script di analisi dati IOT che:**

5) Invia un messaggio di allarme su Profilo Telegram iotclass_bot se si rileva un movimento del sensore (mediante accelerometro o magnetometro)

- Effettuare GET su http://iot2020.cs.unibo.it, porta: 8001
- Parametri: sender, value