UNIVERSITÀ DI PADOVA

FACOLTÀ DI INGEGNERIA

LEWE

DIPARTIMENTO
DI INGEGNERIA
DELL'INFORMAZIONE

Alessandro Pasqualini
A.A. 2015/16
Matr. 1075691

INTERNET OF THINGS

L'espressione "Internet delle cose" indica una famiglia di tecnologie il cui scopo è rendere qualunque tipo di oggetto, anche senza una vocazione digitale, un dispositivo collegato ad internet, in grado di godere di tutte le caratteristiche che hanno gli oggetti nati per utilizzare la Rete.

Davide Bennato

IL PROBLEMA: IL PROTOCOLLO

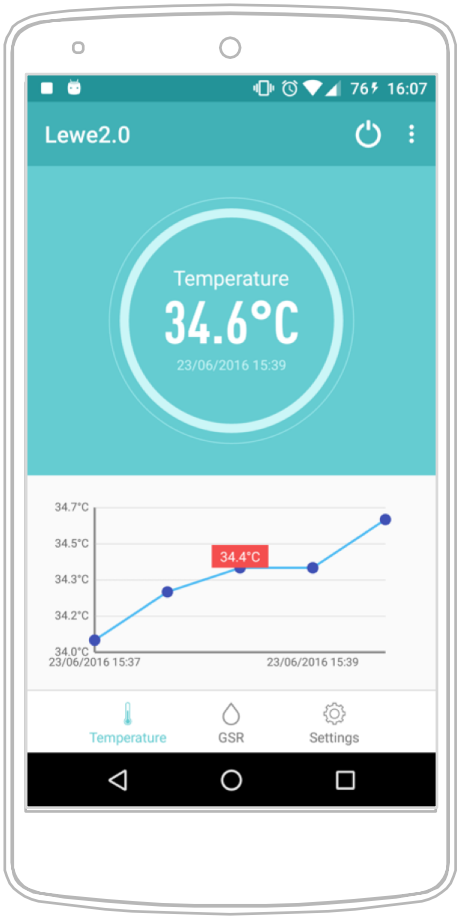
- Mancanza di un comune protocollo di comunicazione
- Predominanza dei protocolli proprietari
- Protocolli non intercompatibili

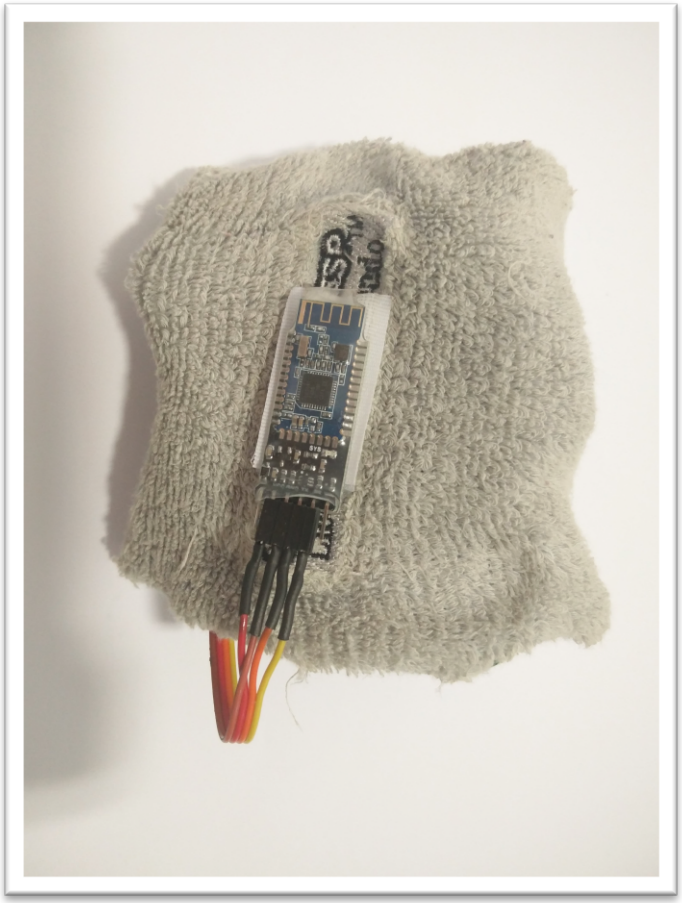
LA SOLUZIONE: IL PROGETTO LEWE

Realizzazione di un protocollo di comunicazione:

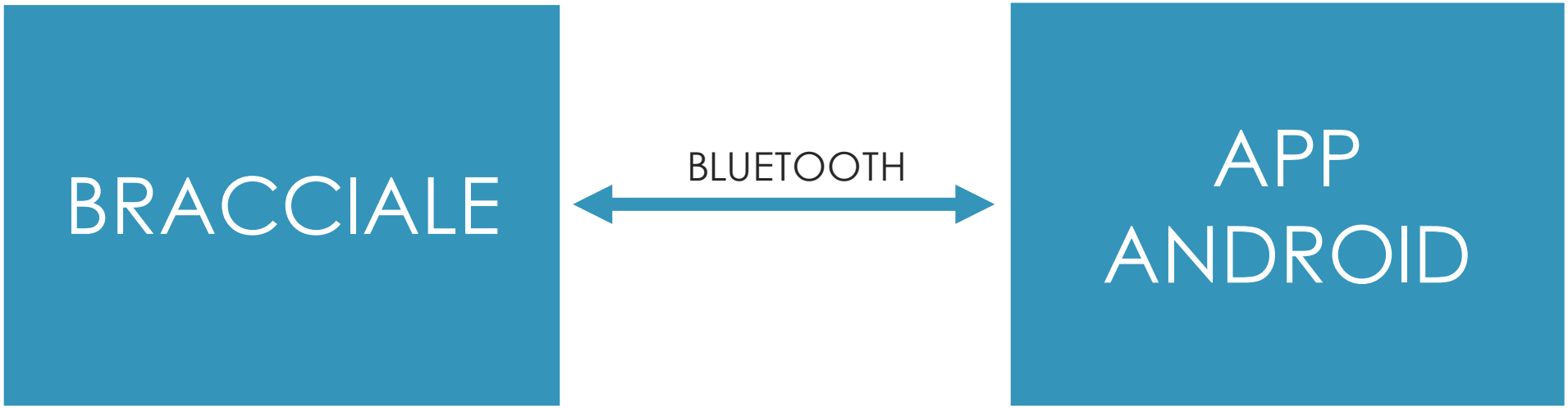
- Leggero
- Flessibile
- Efficiente
- Sicuro

IL CASO DI STUDIO: BRACCIALE BIOMETRICO

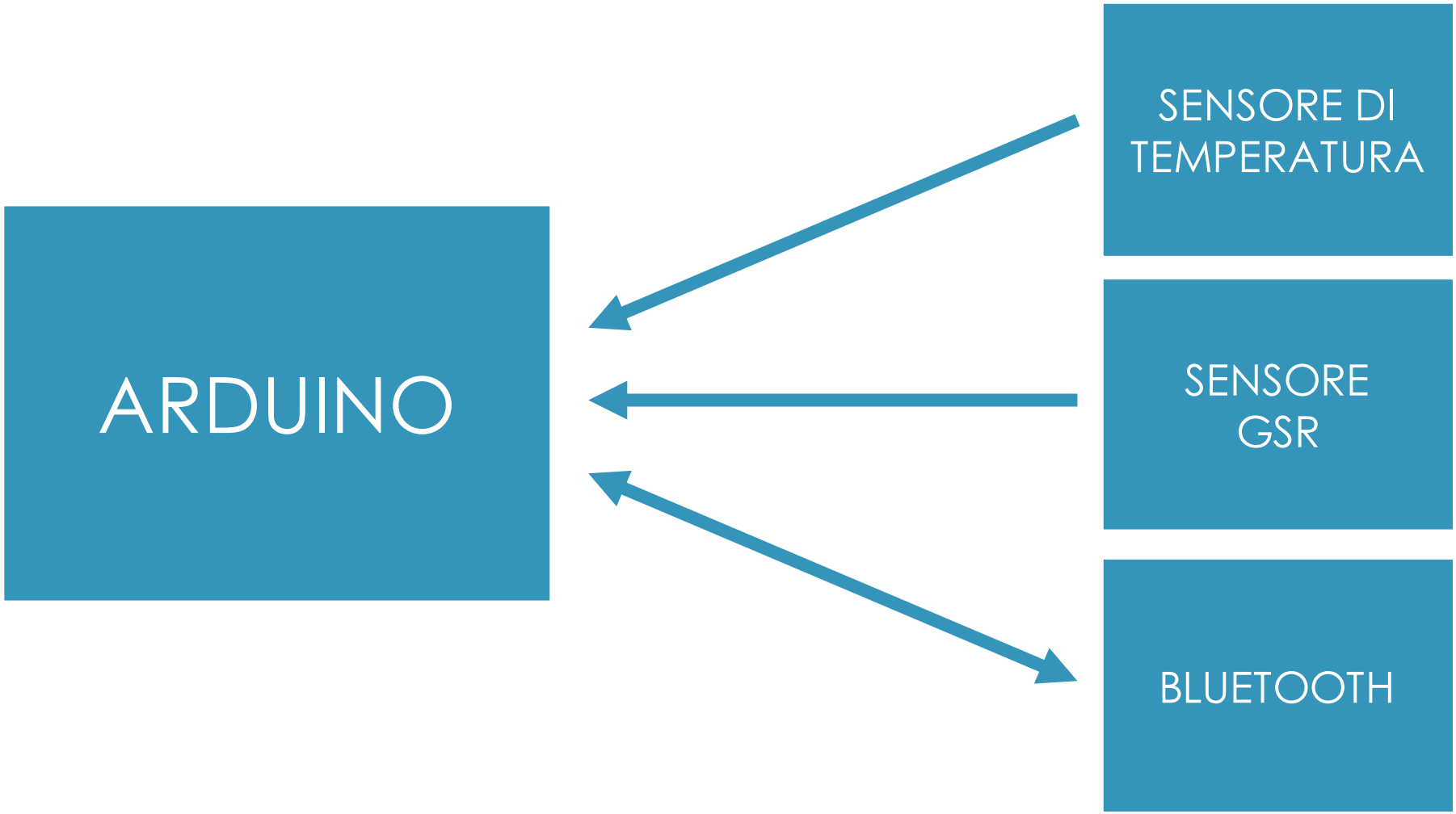




COME E' FATTO?



IL BRACCIALE: HARDWARE



CONNESSIONE BLUETOOTH

ARDUINO

- Comunicazione seriale
- Utilizzo della libreria SoftwareSerial (Arduino IDE)
- Configurazione del modulo HM-10 con comandi AT


```
//creazione oggetto
SoftwareSerial bluetooth(TX, RX);

//inizializzazione
bluetooth.begin(BAUDRATE);

//invio messaggio
bluetooth.print(msg);

//ricezione carattere
char c = bluetooth.read();
```

BLUETOOTH (HM-10)

SENSORE DI TEMPERATURA

Il sensore di temperatura (LM35DZ) fornisce come output 10mV/°C

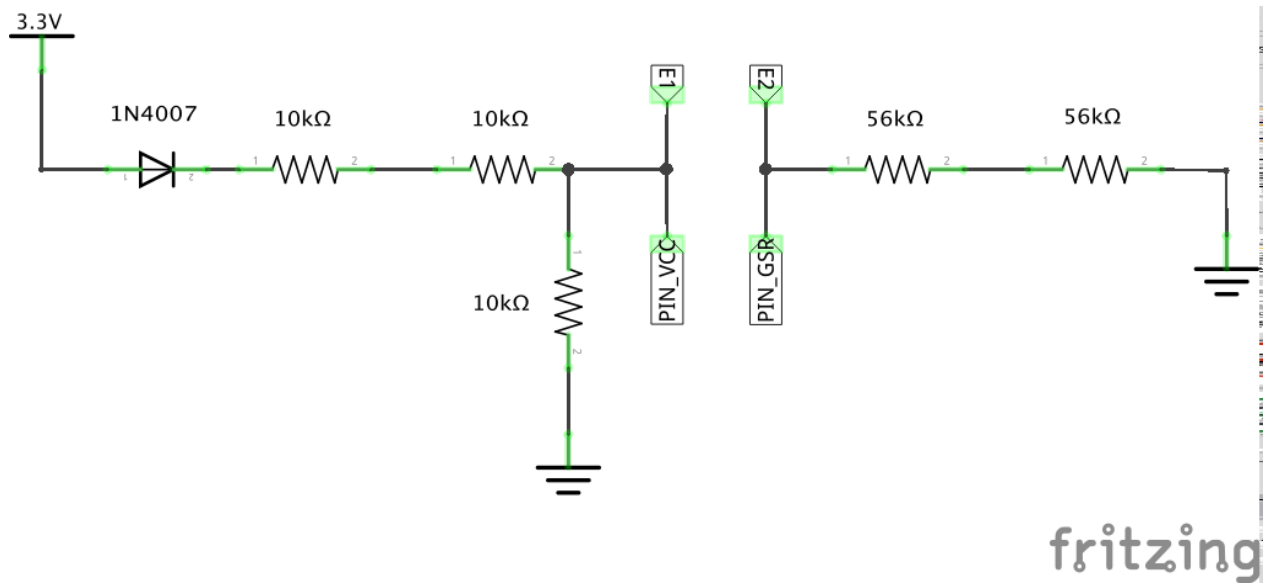
$$Temperatura = \frac{analogRead(PIN_{Temp}) * V_{REF}}{1023} * 100$$

```
double getTemperature() {

   double temp = analogRead(LM35_PIN);
   temp = (temp * VREF / 1023.0) * 100.0;

   double decimalPart = temp - floor(temp);

   temp = floor(temp) + (floor(decimalPart * 10) / 10);

   return temp;

}
```

SENSORE GSR

- Il sensore è formato da due elettrodi a cui è applicata una d.d.p. di 1.1V

- Fornisce in output la caduta di tensione sulla pelle



SENSORE GSR

$$Temperatura = \frac{analogRead(PIN_{Temp}) * V_{REF}}{1023} * 100$$

```
uint8_t getGSR() {

  int gsr = analogRead(GSR_PIN);

  if (gsr > GSR_NOISE) {
    gsr -= GSR_NOISE;
  } else {
    gsr = 0;
  }

  int vcc = analogRead(GSR_VCC_PIN);
  return (uint8_t) (100.0 * gsr / vcc);

}
```

RISPARMIO ENERGETICO

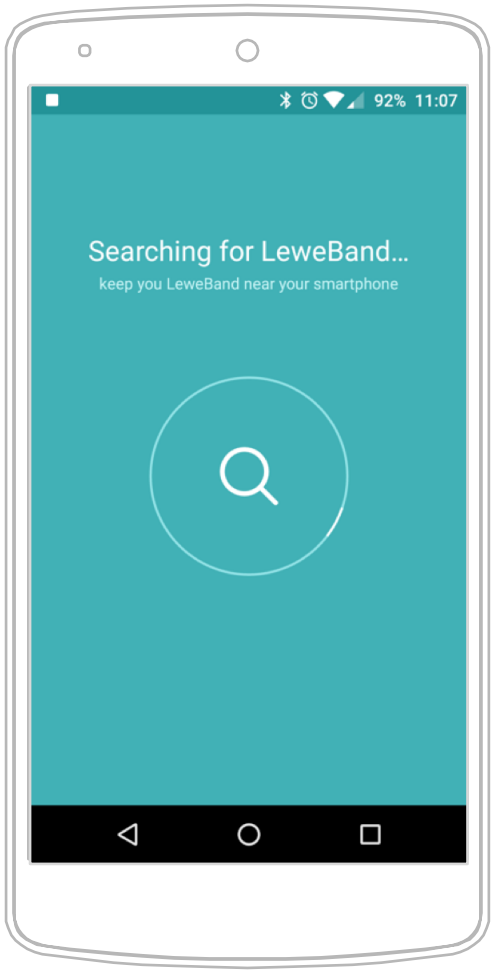
- I sensori vengono accesi solo qualche minuto prima di effettuare la lettura

- Lo stato dei sensori è salvato con l'ausilio del costrutto:

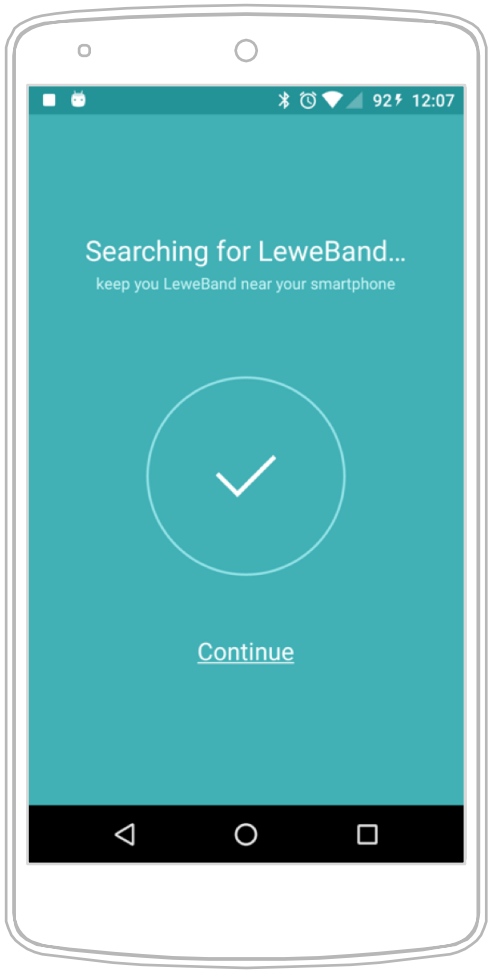
```
typedef enum lwSensorState {
  LW_SENSOR_SLEEP, //sensori addormentati
  LW_SENSOR_WAKE //sensori svegli
};
```

- Sono state realizzate funzioni per pilotare accensione e spegnimento: `wakeupSensor` e `sleepSensor`

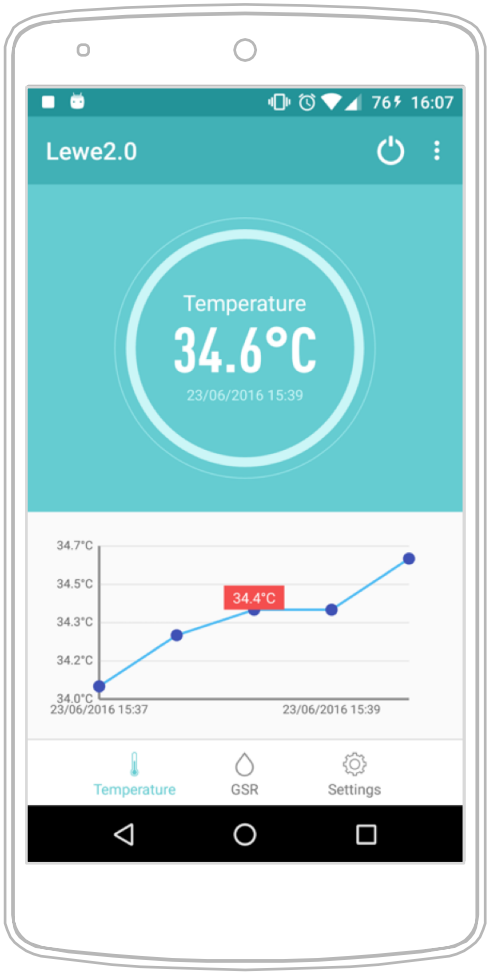
L'APPLICAZIONE ANDROID



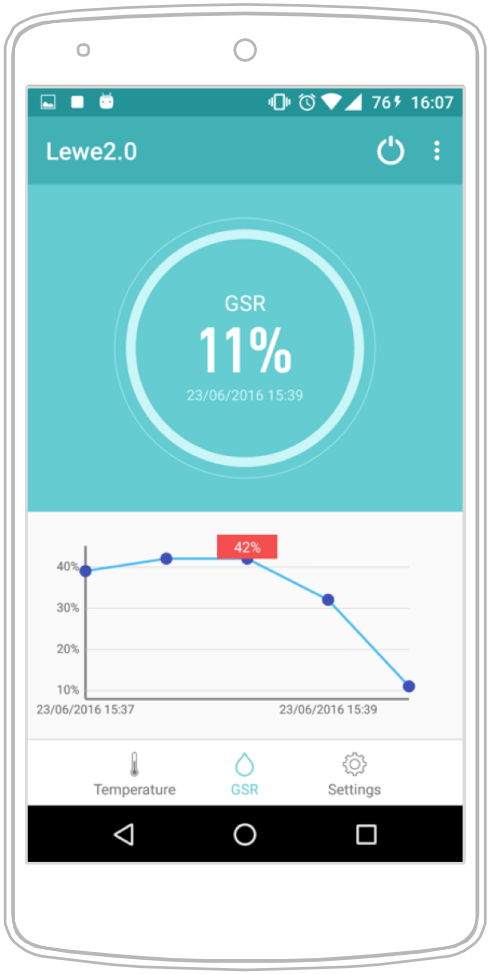
Il processo di pairing



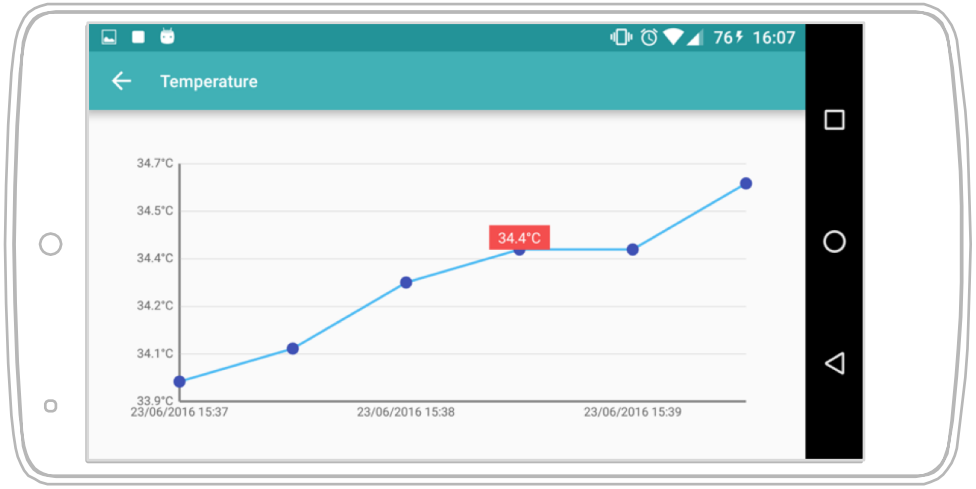
L'APPLICAZIONE ANDROID

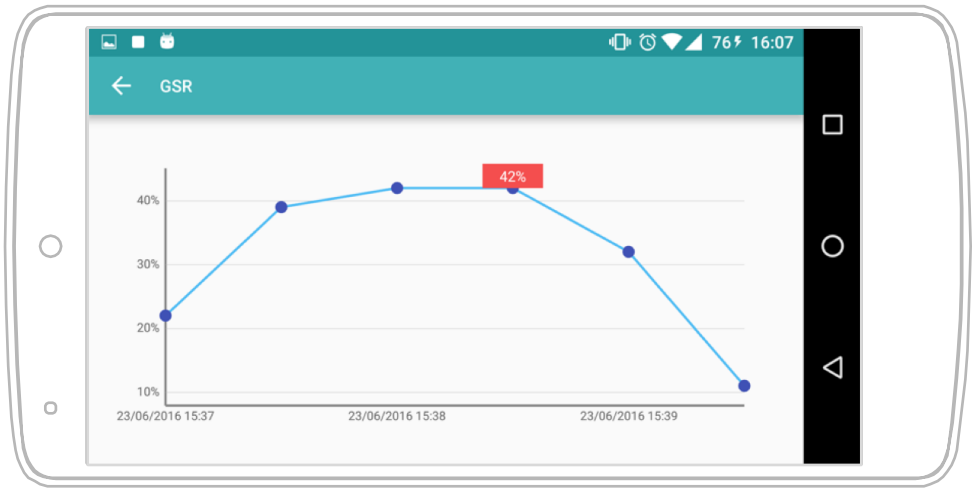


Main activity



L'APPLICAZIONE ANDROID





Grafici di tutte le letture pervenute

IL PROTOCOLLO DI COMUNICAZIONE

Prevede due tipi di messaggi:

- Messaggio DATA
- Messaggio ACK

```
{
    “id”: ID,
    “type”: “data”,
    “val”:
        {
            “key1”: val1,
            “key2: “val2,
        }
}
```

```
{
    “id”: ID,
    “type”: “ack”
}
```

IL PROTOCOLLO DI COMUNICAZIONE

La libreria per Arduino IDE contiene:

- Contenitore dati
- Software di gestione del protocollo
- Software di gestione del canale di trasmissione (bluetooth)

IL CONTENITORE DATI

- E' basato su ArduinoJson (libreria lightweight Json per Arduino
- Prevede un metodo `add` e `get` per ogni tipo primitivo

```
//creazione contenitore
JData contenitore;

//aggiunta dato
contenitore.add("Chiave", "Valore");

//ottenimento dato con autocasting
char *value = contenitore.get("Chiave");
```

SOFTWARE GESTIONE PROTOCOLLO

- Prevede timer per polling del mezzo di trasmissione e per invio automatico dei messaggi
- Metodo **`send`** di invio messaggi

```
long send(JData &message)
```

- Metodo **`loop`** per sincronizzare i timer (simulazione thread)

SOFTWARE GESTIONE PROTOCOLLO

Prevede due eventi:

- Ricezione nuovo messaggio
- Ricezione conferma di un messaggio inviato

Il costruttore

```
Jack(JTransmissionMethod &mmJTM, void (*onReceive)(JData &, long),
void (*onReceiveAck)(long), long (*getMessageID)())
```

prevede il passaggio di due handler (funzioni) per la gestione degli eventi

SOFTWARE DI GESTIONE DEL CANALE DI TRASMISSIONE

- Classe astratta `JTransmissionMethod`

```
virtual size_t receive(char *buffer, size_t size);
virtual void send(char *message, size_t length)
virtual size_t available()
```

- Creata libreria `SoftwareSerialJack` che incapsula un'istanza della libreria `SoftwareSerial` per pilotare il modulo Bluetooth

VALUTAZIONE E COLLAUDO

- Progetto modulare
- Ogni parte è stata testata singolarmente
- Ausilio del preprocessore e della seriale hardware di Arduino

```
#ifdef DEBUG
  Serial.print(F("log"));
#endif
```

VALUTAZIONE E COLLAUDO

- Hardware realizzato preventivamente su breadboard e poi su PCB con l'ausilio di Fritzing
- Per l'app Android sviluppata classe `Logger` che gestisce le informazioni di log

CONCLUSIONI E LAVORO FUTURO

- Json non è adeguato a questo tipo di applicazioni
- Realizzazione di un formato intermedio tra Json e binario
- È necessario ampliare le tipologie dei messaggi (ad esempio messaggi senza conferma)

GRAZIE PER L'ATTENZIONE