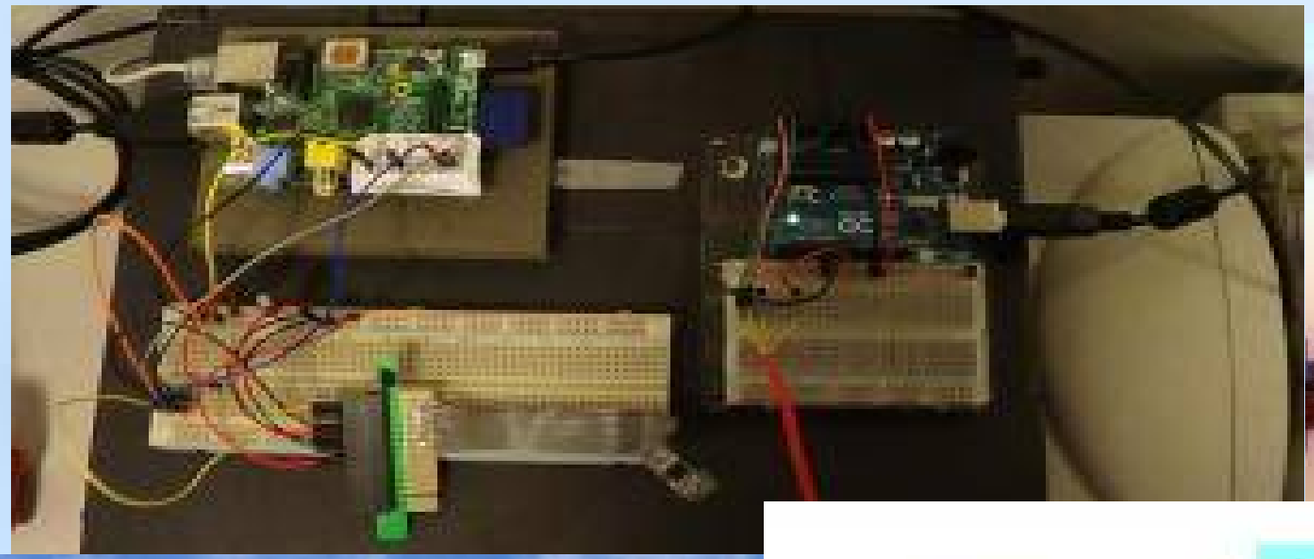
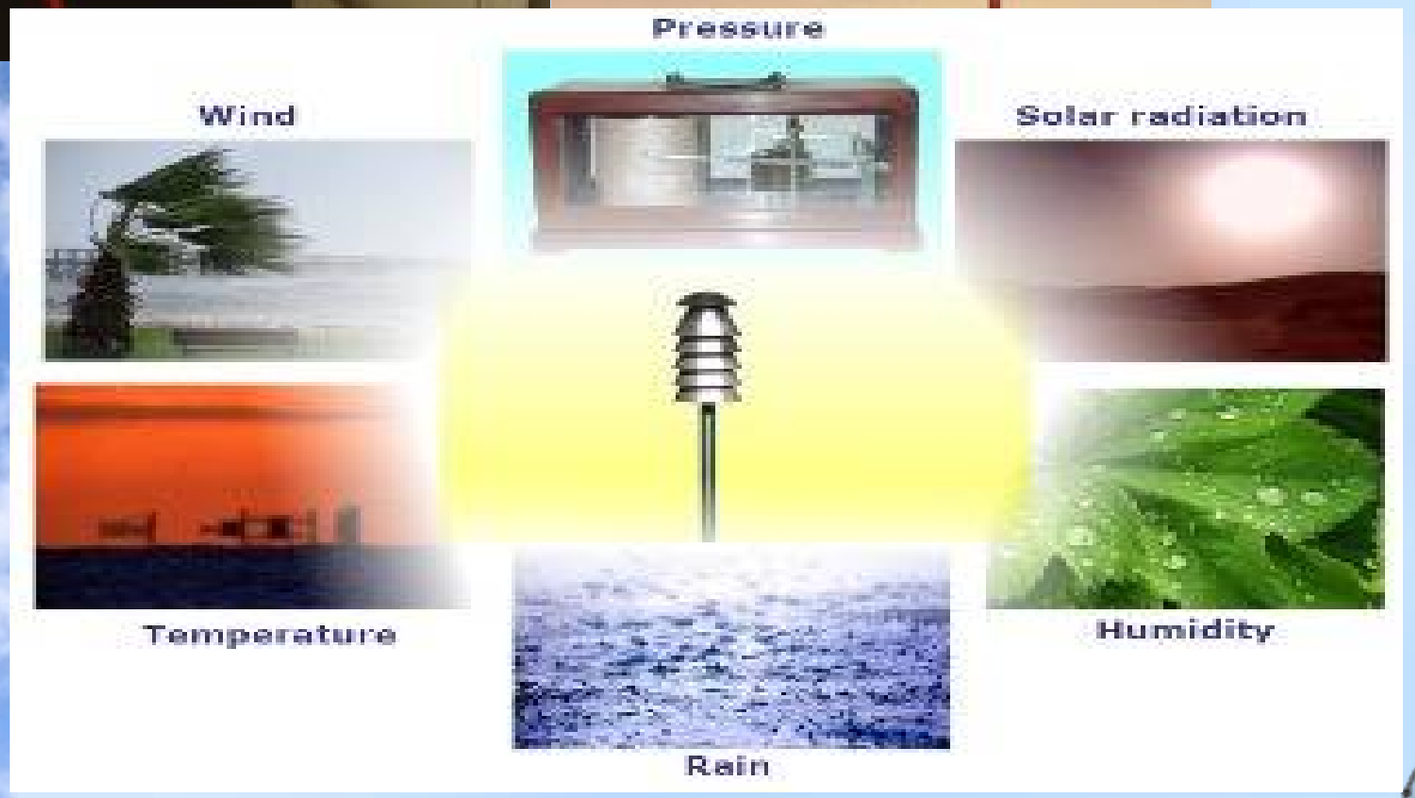
Stima: prototipo di stazione meteo

rete monitoraggio ambientale partecipativo





Smart city, smart citizen e citizen science

- Smart city: territorio urbano che permette di soddisfare le esigenze dei cittadini, delle imprese e delle istituzioni, mediante l'ausilio di strumenti innovativi e partecipazione attiva
- Ad esempio, reti di sensori per il controllo di parametri ambientali.
- Per la Comunità Europea, il grado di intelligenza di una città dovrebbe essere valutato secondo economia, mobilità, ambiente, persone, tenore di vita e governo.
- All'interno di questo ambiente, il cittadino può (deve) avere un ruolo attivo (smart citizen)
- Un esempio di partecipazione attiva può essere ritrovata nella citizen science, una modalità di ricerca scientifica condotta totalmente o in parte da scienziati non professionisti

Inquinamento

- la legislazione europea dice che in una città come Bologna bastano due o tre stazioni per la misura della qualità dell'aria
- sono sufficienti per monitorare gli aspetti generali dell'inquinamento urbano, poi però esistono punti di accumulo locali (sotto i portici, nelle strade strette), differenze tra il primo e l'ultimo piano, emissioni locali (impianti), anche episodiche (cantieri, ingorghi, caminetti accesi), inquinamento dentro le case...
- non basterebbero centinaia di centraline per monitorare tutta questa varietà di situazioni, e i costi delle reti di misura sono già adesso difficili da sostenere
- l'Agenzia Ambientale Europea promuove l'integrazione di strumenti diversi: satelliti, modelli, poche stazioni con strumenti avanzati e reti di microsensori a basso costo gestite da volontari (scuole, associazioni, cittadini)

per un problema multi-scala, monitoraggio multi-scala

- perciò sì, qualche grande pennello e poi pennelli piccoli per definire i dettagli

Obiettivi R-map

- Raccogliere e distribuire dati ambientali rilevati dai cittadini
- Rendere disponibili questi dati ai servizi meteorologici, alle agenzie di prevenzione ambientale, alla protezione civile e istituti di ricerca
- Fornire feedback ai fornitori di dati in modo che essi abbiano servizi per testare e migliorare la qualità dei dati
- Divulgazione scientifica e sensibilizzazione ai temi ambientali
- Coinvolgimento di scuole e università a scopi formativi
- Creare un circolo virtuoso tra Enti Formativi, Pubbliche Amministazioni, Aziende private e cittadini.

Soggetti coinvolti

- **ARPA Emilia Romagna SIMC**
  - Predispone un disciplinare tecnico rispetto ai metodi di misura, elaborazione dei dati dei sensori e loro collocazione
  - Definisce i protocolli e i formati per la comunicazione dei dati
  - Esegue un eventuale controllo di qualità e comunica i risultati
- **ARPA Regione Veneto**
  - Supporta la diffusione degli standard/fornisce infrastruttura hardware
  - Contribuisce allo sviluppo e sperimenta stazioni commerciali con supporto allo standard Rmap
  - Sperimenta nuova sensoristica a basso/medio costo
- **Cineca Consorzio Interuniversitario per il supercalcolo e l'innovazione tecnologica**
  - Sperimenta e sviluppa le tecnologie Stima
  - Supporto BigData
  - Fornisce servizi a valore aggiunto

Soggetti coinvolti

- **Dipartimento informatica Università di Bologna**
  - Esprime pareri sul progetto e prototipo hardware e software
  - Eventuali tesi/tirocini sul progetto software
- **RaspiBO**: gruppo informale di appassionati di elettronica ed informatica libera della zona di Bologna
  - Realizzano un prototipo hardware e software
  - Sperimentano il prototipo
  - Documentano la realizzazione del prototipo come openhardware e distribuiscono il software con licenza libera
- **Soggetto privato / startup**
  - Progetto commerciale per la vendita, installazione e manutenzione delle stazioni
- **Scuole**
- **Soggetti già attivi sul territorio (Meteonetwork)**

Cosa è Rmap

- Un insieme di specifiche:

- **Protocollo di rilevamento dati**
  - Collocazione sensori
  - Accuratezza dei sensori
  - Elaborazioni

  http://www.raspibo.org/wiki/index.php/Gruppo_Meteo/DisciplinareStazione

- **Sistema per lo scambio dati**
  - Protocolli di trasmissione
  - Formati dati
  - Metadati

  http://www.raspibo.org/wiki/index.php/Gruppo_Meteo/RFC-rmap

Implementazioni hardware e software

- Specifiche realizzazioni che aderiscono allo standard Rmap
- Possibilmente open hardware e open software

STIMA/Server R-map

(Acronet)

Specifiche Rmap

Campionamento di variabili meteorologiche

- Campionamento è il processo per ottenere una discretizzata sequenza di misure di una quantità.
- Campione è una singola misura, tipicamente una di una serie di letture “spot” di un sistema di sensoristica.
- Una osservazione (misurazione) è il risultato del processo di campionamento. Nel contesto di analisi di serie, un'osservazione è derivato da un numero di campioni.

- Variabili atmosferiche come la velocità del vento, temperatura, pressione e umidità sono funzioni di quattro dimensioni - due orizzontali, una verticale e una temporale. Esse variano irregolarmente in tutte e quattro, e lo scopo dello studio del campionamento è quello di definire le procedure di misura pratiche per ottenere osservazioni rappresentative con incertezze accettabili nelle stime delle medie e variabilità.

Data Level

- Dati **Level I** , sono le letture dirette degli strumenti espresse in appropriate unità fisiche e georeferenziate

- Dati **Level II**, dati riconosciuti come variabili meteorologiche; possono essere ottenuti direttamente da strumenti o derivati dai dati Level I

- Dati **Level III** sono quelli contenuti in dataset internamente consistenti, generalmente su grigliato.

- I dati scambiati a livello internazionale sono livello II o livello III

Disciplinare per il rilevamento di dati

Per ora una proposta per:

- **Schermi dalla radiazione**
- **Temperatura**
- **umidità**

Prima prebozza disponibile a:

- http://www.raspibo.org/wiki/index.php/Gruppo_Meteo/DisciplinareStazione

Protocolli per R-map

- **MQTT** (Message Queue Telemetry Transport) è un protocollo publish/subscribe particolarmente leggero, adatto per la comunicazione M2M tra dispositivi con poca memoria o potenza di calcolo e server o message broker.
- **AMQP** (Advanced Message Queuing Protocol) è protocollo per comunicazioni attraverso code di messaggi. Sono garantite l'interoperabilità, la sicurezza, l'affidabilità, la persistenza. Nella sua implementazione Rabbitmq exporta un broker MQTT e fornisce delle api web
- Json è il formato per il payload

E' fondamentale:

- Integrazione con le funzioni e le specifiche richieste dalle applicazioni per la **domotica**
- Integrazione con applicazioni per la **telefonia mobile** per la rilevazione dello spessore neve e altri parametri

Conceptual data model

These models, sometimes called domain models, are typically used to explore domain concepts with project stakeholders

DB-All.e Conceptual data model

- Il modello è orientato all'applicazione (bisogna capire cosa sono i dati, normalizzarli e ricondurli a metadati stardard in fase di accoglienza), quindi si lavora pre e non post
- I dati sono legati ai metadati in modo univoco
- Una osservazione è univoca nello spazio dei suoi metadati
- L'unica possibilità di far coesistere due osservazioni dello stesso parametro nello stesso punto è atraverso il metadato “network” associabile alla classe dello strumento
- La tracciabilità di un sensore, una stazione, un osservatore nello spazio, nel tempo etc. Avviene attraverso il metadato “ident”
- Alcuni metadati sono table driven (level,timerange,network)

- Ogni dato può essere associato a un certo numero di attributi
- Nessuna dimensione è vincolata (intervalli temporali tra dati, numero attributi....)
- E' contemplata la gestione di previsioni; il datetime è sempre quello di verifica
- Misure e metadati hanno troncamenti sulle cifre significative dettati dalla loro possibilità reale di misura e stabiliti a priori
- Esistono due categorie di dato: una che varia tutti i metadati (osservazioni classiche) e l'altra che non prevede l'uso di alcuni metadati e che quindi sono da considerarsi come ulteriori metadati di quella singola stazione (constant station data: es. Nome stazione)
- Nessuno vieta di espandere i metadati estendendo esternamente questo data model

Logical data model (LDM)

LDMs are used to explore the domain concepts, and their relationships, of your problem domain

DB-All.e LDM

- ***METADATI***
  - ***Datetime****: tempo di fine misurazione*
  - *Ana:* ***Longitudine****,* ***latitudine*** ed un ***identificativo***
  - ***network***: definisce stazioni con caratteristiche omogenee (classe degli strumenti)
  - ***Time range***: Tr,P1,P2 indica osservazione o tempo previsione ed eventuale elaborazione “statistica”
  - ***Level***: TL1,L1,TL2,L2 le coordinate verticali (eventualmente strato)
  - ***Variable****: Btable parametro fisico*
- ***DATI***
  - *Valori rappresentabili come interi, reali, doppia precisione, stringhe*
    - ***Attributi*** *(alla stregua di dati)*

Metadati su MQTT

- Ogni topic corrisponde ai metadati univoci, mentre il payload è composto dal valore e dall'instante temporale
- **/IDENT/COORDS/NETWORK/TRANGE/LEVEL/VAR**
  - **IDENT**: identificativo per stazioni mobili, “-” per stazioni fisse
  - **COORDS**: nella forma *lon,lat*. Le coordinate sono espresse nell forma *int(valore*10^5)* con eventuale segno negativo
  - **NETWORK**: massimo 16 caratteri
  - **TRANGE**: nella forma *indicator,p1,p2*
    - *Indicator* e *p2* interi senza segno, *p1* intero con eventuale segno negativo. “-” per valori non significativi
  - **LEVEL**: nella forma *type1,l1,type2,l2*
    - *Type1, type2* interi con eventuale segno negativo, *l1*e *l2* interi con eventuale segno negativo. “-” per valori non significativi
  - **VAR**: nella forma *BXXYYY*
- Il payload è in formato JSON: *{ “v”: VALUE, “t”: TIME, “a”: { “BXXYYY”: VALUE, … } }*
  - *VALUE*: valore in formato CREX
  - *TIME:* formato YYYY-mm-ddTHH:MM:SS.MSC (secondi e millisecondi opzionali)
  - Gli attributi (*“a”)* sono opzionali

RMAP web services

Composizione degli URL per un HTTP GET request

La "base" della richiesta è quella standard:

/version/ident/coords/network/timerange/level/bcode/

Ad esempio:

http://rmap.cc/v0.1/-/1207738,4460016/locali/
254,0,0/103,2000,-,-/B12101

Serie temporale

Serie temporale mensile, giornaliera e annuale:

/ident/coords/network/timerange/level/bcode/timeseries/year

/ident/coords/network/timerange/level/bcode/timeseries/year/month

/ident/coords/network/timerange/level/bcode/timeseries/year/month/day

Ad esempio:

/-/1207738,4460016/locali/254,0,0/103,2000,-,-/B12101/timeseries/2011

/-/1207738,4460016/locali/254,0,0/103,2000,-,-/B12101/timeseries/2011/01

/-/1207738,4460016/locali/254,0,0/103,2000,-,-/B12101/timeseries/2011/01/13

Serie spaziale e sommario

Serie spaziale di una rete, con granularità oraria (± 30 minuti):

/*/*/NETWORK/TIMERANGE/LEVEL/BCODE/spatialseries/YEAR/MONTH/DAY/HOUR

Riassuntivo

/*/*/*/*/*/*/summaries

/*/*/*/*/*/*/summaries

- /*/*/NETWORK/*/*/*/summaries
- /-/COORDINATES/NETWORK/*/*/*/summaries
- /IDENT/*/NETWORK/*/*/*/summaries
- /*/*/NETWORK/TIMERANGE/LEVEL/BCODE/summaries/YEAR/MONTH
- /*/*/NETWORK/TIMERANGE/LEVEL/BCODE/summaries/YEAR/MONTH/DAY

Bufr

- The Binary Universal Form for the Representation of meteorological data (BUFR) is a binary data format maintained by the World Meteorological Organization (WMO).
- BUFR was designed to be portable, compact, and universal. Any kind of data can be represented, along with its specific spatial/temporal context and any other associated metadata. In the WMO terminology, BUFR belongs to the category of table-driven code forms, where the meaning of data elements is determined by referring to a set of tables that are kept and maintained separately from the message itself.
- Descriptors: all element descriptors will be found in BUFR specification section known as "Table B". The Table B definition of an element descriptor includes its number, short text definition, decoding parameters (bit width, scale factor, and bias), and type (numerical, character string, code table, etc.).

Software di decodifica Bufr

- **WREPORT**: a featureful C++ library for BUFR and CREX encoding and decoding
  http://sourceforge.net/p/wreport/home/Home/

- **GRIBAPI ecCodes**
  https://software.ecmwf.int/wiki/display/ECC/ecCodes+Home

Formato JSON

- http://www.json.org/json-it.html

- JSON (JavaScript Object Notation) è un semplice formato per lo scambio di dati. Per le persone è facile da leggere e scrivere, mentre per le macchine risulta facile da generare e analizzarne la sintassi.
- Rispetta il data model che ci siamo dati
- Ogni elemento è il report con i dati di una certa stazione per un certo istante di riferimento.
- Fornito mediante postoprocessatore **json** di Arkimet, ma disponibile anche come tool da riga di comando

Formato GeoJSON per punti sparsi

- http://geojson.org/
- E' un formato molto verboso (soprattutto per le serie temporali) ma
- Rispetta il data model che ci siamo dati (anche se in modo non ottimale)
- E' uno dei formati vettoriali interpretati da **GDAL/OGR**
  - OpenLayers
  - QGIS
  - ...
- Fornito mediante postoprocessatore **json** di Arkimet, ma disponibile anche come tool da riga di comando

Il prototipo si stazione sviluppato da RaspiBO

I principi dell'Open Source Hardware (OSHW)

- **hardware** il cui progetto è reso **pubblico** in modo che chiunque possa **studiare**, **modificare**, **distribuire**, **realizzare**, e **vendere** il progetto o l'hardware basato su di esso
- La fonte dell'hardware, il progetto da cui è stato realizzato, è disponibile nel **formato migliore per apportarvi modifiche.** Idealmente, l'hardware open source **utilizza componenti e materiali disponibili, processi standard, infrastruttura aperta, contenuti senza restrizione e strumenti di progettazione open-source** per massimizzare la capacità degli individui di produrre e utilizzare l'hardware.
- L'hardware open source dà alle persone la libertà di controllare la loro tecnologia, la condivisione della conoscenza ed incoraggia il commercio attraverso lo scambio aperto di progetti.

Open Source Hardware Association
http://www.oshwa.org/definition/italian/

La Definizione 1.0 dell'Open Source Hardware

- La documentazione :La documentazione deve includere i file del progetto nel formato preferito per apportare modifiche, ad esempio, il formato nativo del file di un programma CAD
- Il Software necessario:
  - Le interfacce sono sufficientemente documentate tale che si possa scrivere il software open source che consente al dispositivo di funzionare
  - Il software necessario è rilasciato sotto una licenza open source
- I lavori derivati: permettere modifiche e lavori derivati
- La ridistribuzione libera
- L'attribuzione: riportare l'attribuzione ai licenzianti quando si fa la distribuzione; può richiedere che i lavori derivati abbiano un nome o un numero di versione diversi dal progetto originale
- Nessuna discriminazione di persone o gruppi
- La licenza non deve essere specifica per un prodotto

STIMA: vista d'insieme

- **Open source hardware e software**
- **Utilizzo delle piattaforme più diffuse e board di prototipizzazione**
- **Disegni hardware con Kicad (by Daniele Castellari)**
- **5 moduli hardware che soddisfano differenti esigenze**
  - **Consumi**: alimentazione tramite rete, batterie con pannello solare, batterie
  - **Collegamento**: all'interno della casa, in esterno con un cavo ethernet e PoE, in esterno in postazione fissa ravvicinata, in esterno in postazione mobile
  - **Domotica**: sono collegabili attuatori
- Un modulo con funzioni **server**
  - Database
  - Web server
  - NTP server
  - Sviluppo

E' anche un “framework” per makers

- E' possibile utilizzare il modulo base per lo sviluppo del firmware
- Sono installate le librerie Arduino con i file di specifiche per i microcontrollori utilizzati
- Come build system si può utilizzare ARDUINO
- Tutte le librerie personalizzate sono già installate
- L'aggiornamento avviene tramite git / pip
- E' possibile “scriptare” il build, upload e configurazione delle board

Modulo base: vista d'insieme

circa 75€

- Sviluppato su **Raspberry**

- Distribuzione Pidora 2014 (fedora 20); migrazione a Centos 7
- Software completamente pacchettizzato RPM
- Repository software pubblico

  http://rmapv.rmap.cc/repo/rmap/fedora/20/RPMS/arm/repoview/

- Per ora una immagine SD da 8G scaricabile e pronta all'uso
- Il modulo gestisce direttamente la sensoristica su I2C
- Testata connessione alla LAN con Ethernet WIFI e GSM
- Gestione di una propria LAN con dhcp server, dsn server e nat

Trasporto

Il concetto di trasporto in Stima è simile ma non rigidamente aderente ai concetti del modello ISO-OSI.

Nel caso dei trasporti passivi il suo compito è fornire un canale logico-affidabile di comunicazione end-to-end per fornire servizi al soprastante livello che in Stima è JsonRPC.

Nel caso dei trasporti attivi corrisponde al protocollo (Session Layer) per la pubblicazione dei dati su un server (broker).

Trsporti passivi

In pratica i trasporti "passivi" permettono di eseguire procedure remote codificate in formato json specifiche dell'implementazione Stima; quelli attivi permettono la pubblicazione su server (broker) dei messaggi aderenti allo standard R-MAP.

Trasporti Passivi:

- Seriale
- TCP/IP
- Bluetooth (serial port profile)
- NRF24

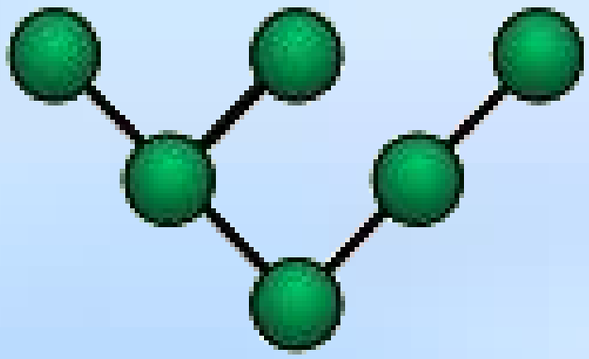
Trsporti attivi

Trasporti attivi:

- MQTT
- AMQP

Trasporti

- Trasporto **Bluetooth** (HC-05)
- Trasporto **Seriale**
  - Principalmente per configurazione e debug
  - Piccole distanze via cavo
- Trasporto **TCP/IP**
  - Collegamenti tramite cavo ethernet a breve e media distanza
- Trasporto **RF24Network**
  - OSI Network Layer using nRF24L01(+) radios 2.4GHz ISM
  - 50/150m in aria libera
  - Host Addressing. Each node has a logical address on the local network.
  - Message Forwarding. Messages can be sent from one node to any other, and this layer will get them there no matter how many hops it takes.
  - Ad-hoc Joining. A node can join a network without any changes to any existing nodes.



Tree

RF24Network Addressing and Topology

Each node must be assigned an 15-bit address by the administrator. This address exactly describes the position of the node within the tree. The address is an octal number. Each digit in the address represents a position in the tree further from the base.

- Node 00 is the base node.
- Nodes 01-05 are nodes whose parent is the base.
- Node 021 is the second child of node 01.
- Node 0321 is the third child of node 021, an so on.
- The largest node address is 05555, so 3,125 nodes are allowed on a single channel.

- Alla libreria distributia è stata aggiunta la crittografia e frammentazione e ricomposizione del payload

Per ora 4 tipi di trasporto

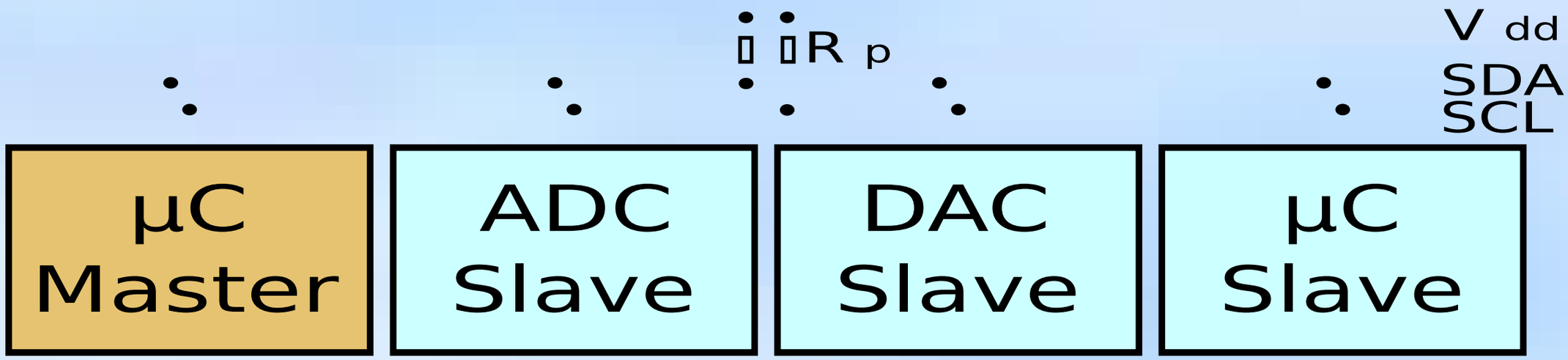
I moduli possono essere così caratterizzati dal trasporto supportato, se eseguono RPC, se richiedono RPC, se pubblicano su MQTT.

I moduli prototipati in r-map sono così denominati e caratterizzati:

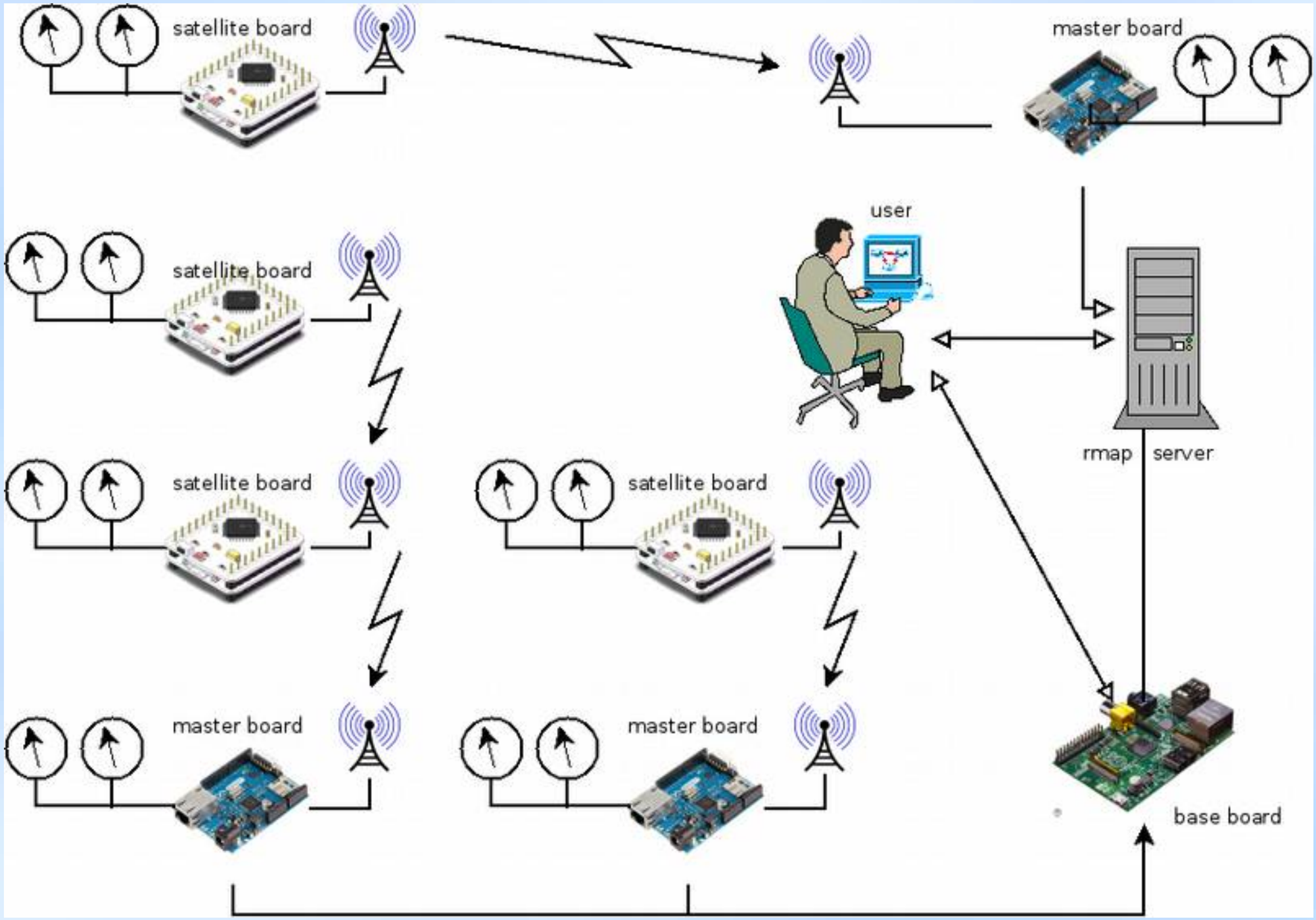
	Attivo / Passivo	Trasp: Seriale	Trasp: TCP/IP	Trasp: radio RF24 Network	pubblish on MQTT	funzioni server	Alimentazione
Modulo Base	Attivo	Si	Si	Da sviluppare	Si	Si	Rete Batterie con pannello solare
Modulo Bluetooth	Passivo	Si	No	Si	No	No	Batterie
Modulo master	Attivo / Passivo	Si	Si	Si	Si	No	Ethernet PoE
Modulo satellite	Passivo	Si	No	Si	No	No	Batterie (con pannello solare)
Modulo GSM	Attivo/Passivo	Si	(Si)	Si	Si	No	Batterie con pannello solare

BUS I2C

- Il protocollo i2c prevede l'utilizzo di un bus formato da due linee bidirezionali. Le due linee, chiamate "scl" e "sda" rispettivamente, trasportano la tempistica di sincronizzazione (chiamata anche "clock") e i dati.
- Abbiamo scelto il bus i2c in quanto:
  - È diventato lo standard di fatto per una serie di integrati tra cui i sensori
  - Si possono collegare fino a 127 dispositivi
  - La comunicazione è bidirezionale (read e write) con velocità assolutamente sufficienti per i nostri scopi



Schema comunicazione hardware



Json-rpc

- Json
  - JavaScript Object Notation ed è un formato adatto ad immagazzinare varie tipologie di informazioni, e quindi a scambiare queste informazioni tra applicazioni client/server.
  - JSON possiede una struttura semplicissima

- JSON-RPC is lightweight remote procedure call protocol similar to XML-RPC. It's designed to be simple!

  Esempi:

--> {"jsonrpc": "2.0", "method": "subtract", "params": {"subtrahend": 23, "minuend": 42}, "id": 3}

<-- {"jsonrpc": "2.0", "result": 19, "id": 3}

--> {"jsonrpc": "2.0", "method": "subtract", "params": {"minuend": 42, "subtrahend": 23}, "id": 4}

<-- {"jsonrpc": "2.0", "result": 19, "id": 4}

Jsonrpc: la richiesta

Tutti i parametri trasferiti di ogni tipo sono singoli oggetti, serializzati usando JSON. Una richiesta è una chiamata a uno specifico metodo disponibile sul sistema remoto; deve contenere tre specifiche proprietà:

- method - Una stringa col nome del metodo da invocare.
- Params - Un array di oggetti come parametri al metodo invocato.
- id - Un valore di qualsiasi tipo, usato per riferire la risposta alla richiesta a cui si sta rispondendo.

Jsonrpc: la risposta

Il server che riceve la richiesta deve rispondere con una risposta valida a tutte le richieste ricevute. Una risposta deve contenere le proprietà descritte qui sotto:

- result - I dati ritornati dal metodo invocato. Se c'è un errore invocando il metodo, il valore deve essere null.
- error - Uno specifico codice di errore se l'invocazione del metodo ha dato luogo a un errore, altrimenti null.
- id - L'id della richiesta a cui si sta rispondendo.

Json-rpc un modo per fare tutto...

Questo un esempio di interrogazione e risposta di un sensore di temperatura

- SEND:{"jsonrpc":"2.0", "method":"getjson", "params":{"node":1, "type":"TMP", "driver":"I2C", "address":72},"id": 0}
- RECEIVE: {"jsonrpc":"2.0","result":{"B12101":30633},"id":0}

B12101 indica che il numero che segue è una temperatura in centesimi di gradi Kelvin, quindi 33.18 C.

Le remote procedure disponibili sono documentate sul wiki del progetto: http://www.raspibo.org/wiki/index.php/Gruppo_Meteo/RemoteProcedure

MQTT (Message Queue Telemetry Transport)

- è un protocollo publish/subscribe particolarmente leggero, adatto per la comunicazione M2M attraverso un tramite detto broker

- Il mittente di un messaggio si limita a "pubblicare" il proprio messaggio al broker. I destinatari si rivolgono a loro volta al broker "abbonandosi" alla ricezione di messaggi.

- Il meccanismo di sottoscrizione consente ai subscriber di precisare a quali messaggi sono interessati tramite un pattern (topic).

- Client e broker si scambiano messaggi di polling per monitorare lo stato delle comunicazioni; sono previsti messaggi "will and testament"

Caratteristiche moduli GSM, Master, Satellite e Bluetooth

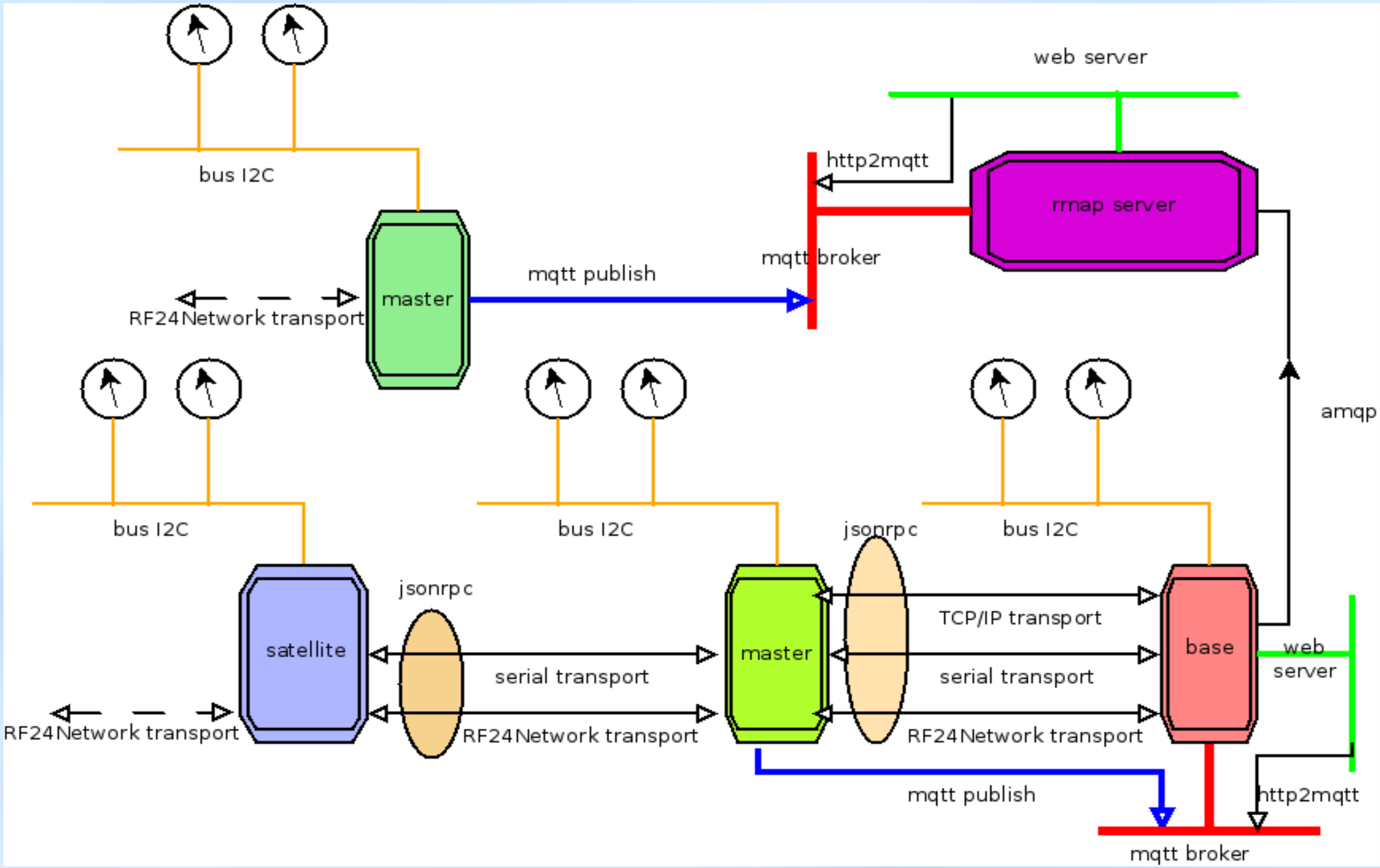
- Firmware unico configurabile al tempo della compilazione
- Parametri di configurazione run time salvati su EEPROM
- Debug on serial insieme a json-rpc
- WatchDog con timeout di 8 sec
- Modalità sleep con interrupt sulla radio per modulo satellite
- RTC sincronizzato tramite NTP / GPS / http

Gestione dei sensori

Libreria di "driver" per sensori

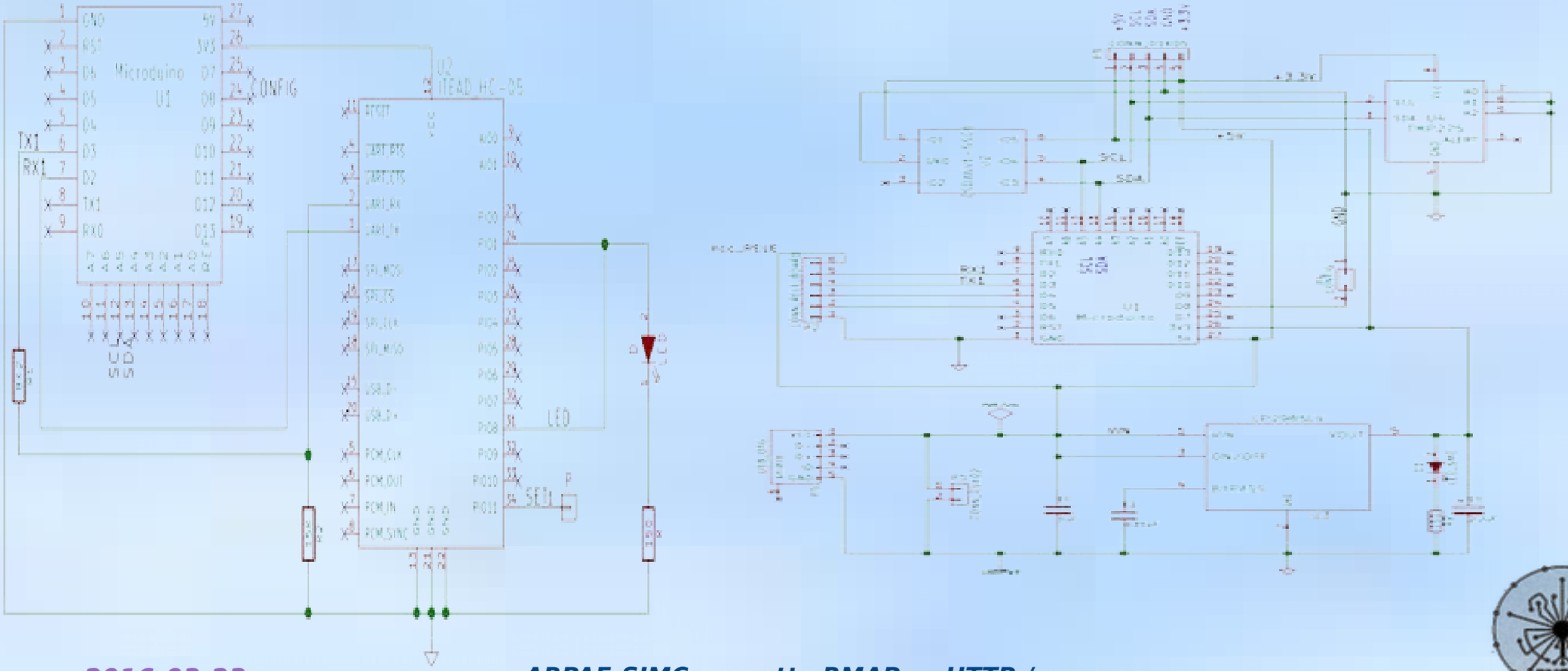
- Esistono attualmente due versioni, una in C++ e una in python
- Porta le gestione della sensoristica ad un livello di astrazione più alto. Aggiungere un nuovo tipo di sensore consiste nell'estendere una classe con quattro metodi per effettuare la lettura di quello specifico sensore:
  - int **setup**(int address);effettua eventuali settaggi necessari al funzionamento del sensore; esempio per temperatura: numero di bit di risoluzione, operazione di misura one-shot
  - int **prepare**(unsigned long* waittime); impartisce al sensore il comando per effettuare una singola misurazione torna il tempo in millisecondi di attesa necessario
  - int **get**(int* value); torna i valori della misurazione
  - aJsonObject* **getJson**() = 0; torna i valori in formato json

Schema comunicazione tra moduli

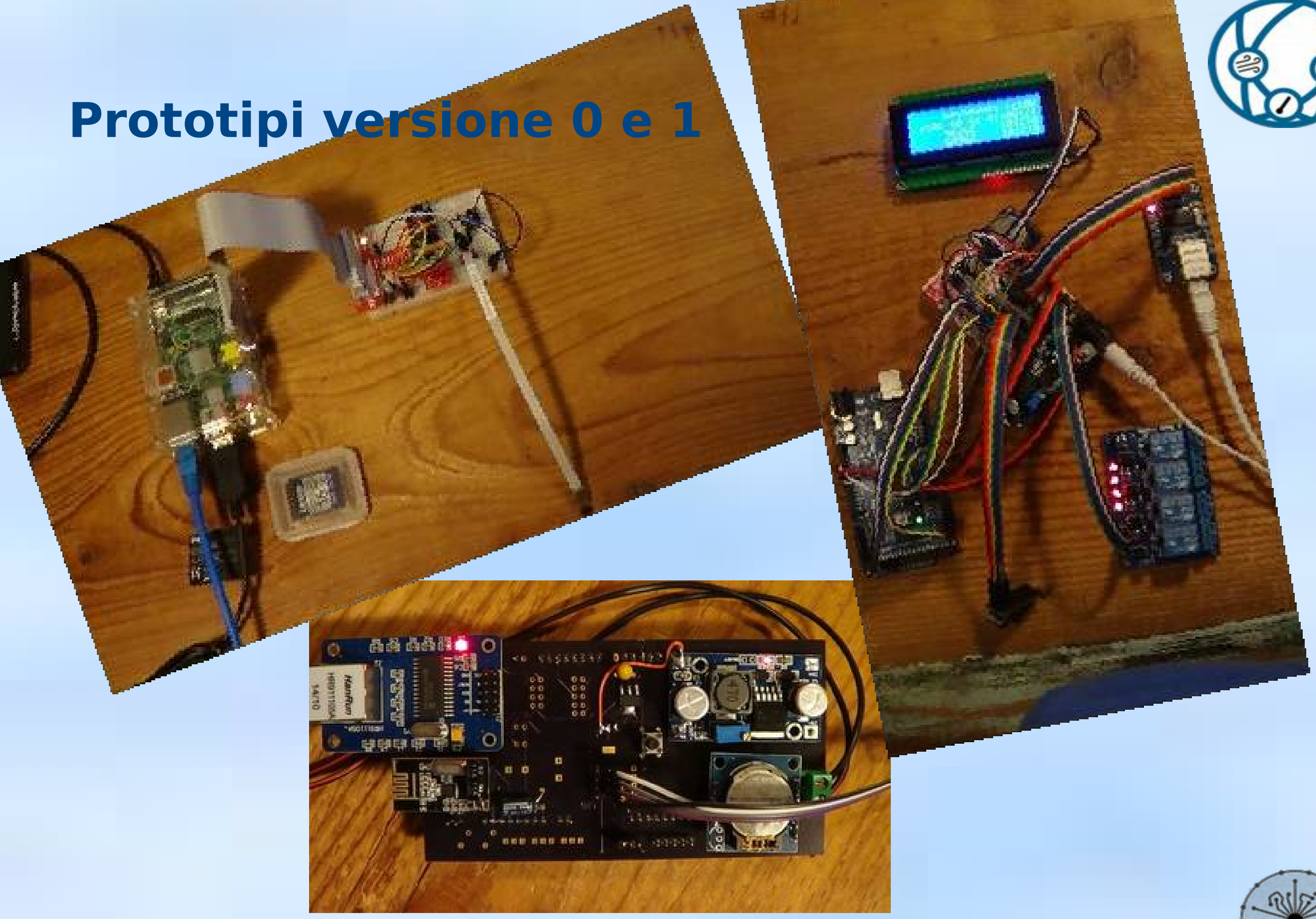


KiCad

- KiCad è una suite open source di software Electronic Design Automation (EDA) per il disegno di schemi elettrici e circuiti stampati (PCB). Ha un ambiente di sviluppo integrato (IDE) con editor di schematici, generazione della distinta base, sbroglio circuitale del PCB e visualizzatore di file Gerber.



Prototipi versione 0 e 1



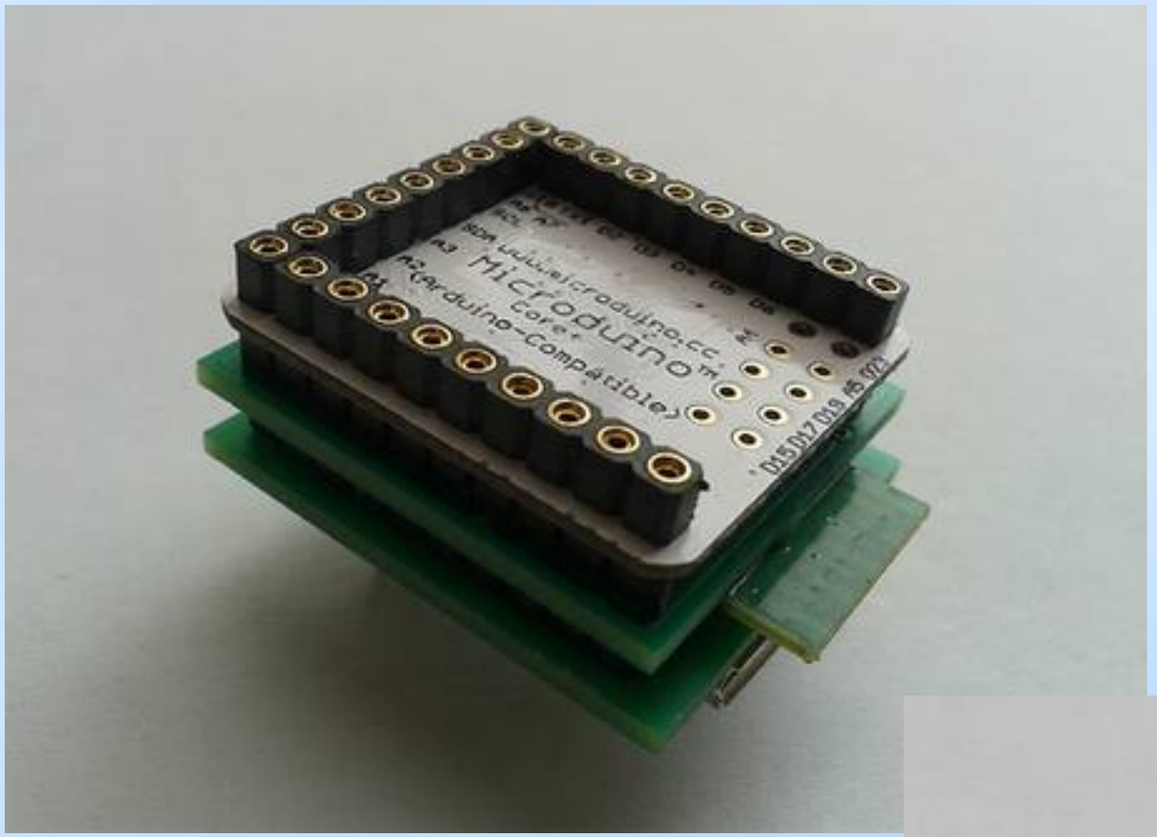
Board con microcontroller versione 1
circa 50€ /100€

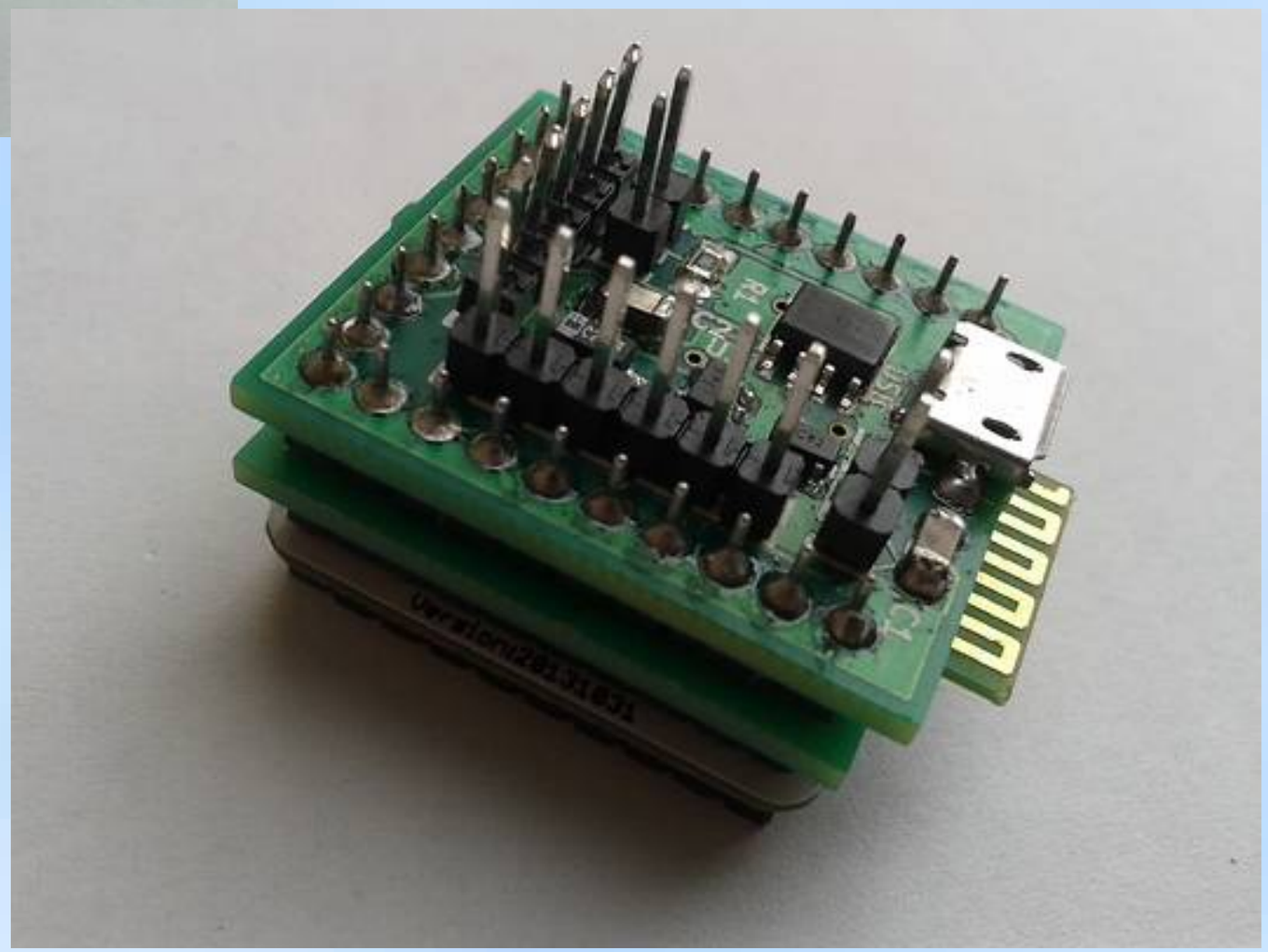
- Modulo master
  - Microduino core+ 1284p / arduino mega 2560
  - Breadborad/circuito stampato
  - scheda RTC
  - scheda mini ethernet ENC60 /microduino ENC / scheda SIM900
  - scheda radio RF24
  - cavo power over ethernet
  - stabilizzatore dcdc switched
  - sensori
- Modulo satellite Modulo Bluetooth
  - microduino core+ 644p @5V 16MHz
  - microduino RF24 con antenna / scheda Bluetooth
  - microduino seriale USB FT232R
  - sensori

Modulo Stima-Bluetooth

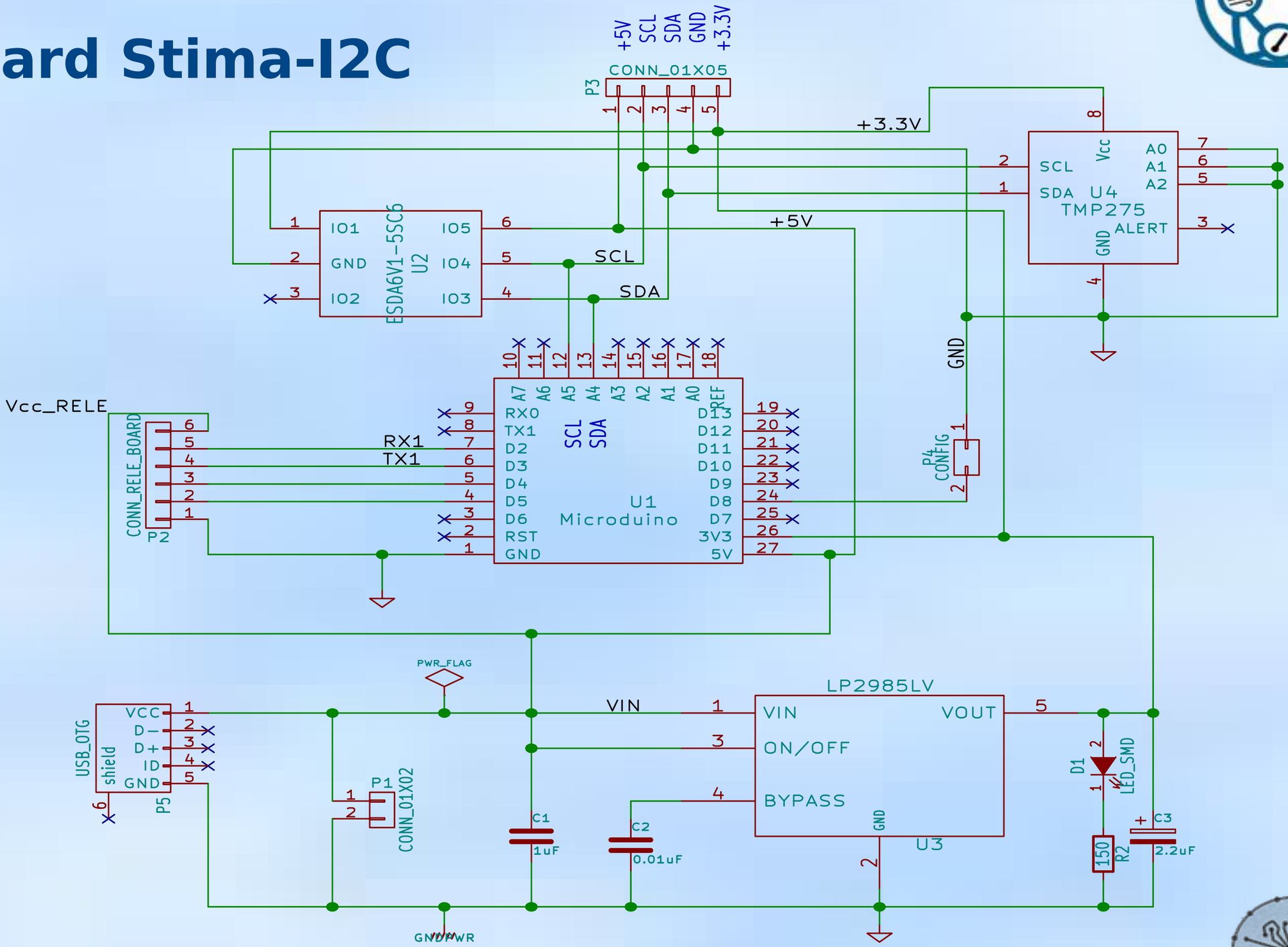
Il modulo è composto da tre board:

- **board microduino core+**
- **board stima-I2C**
- **board stima-bluetooth**

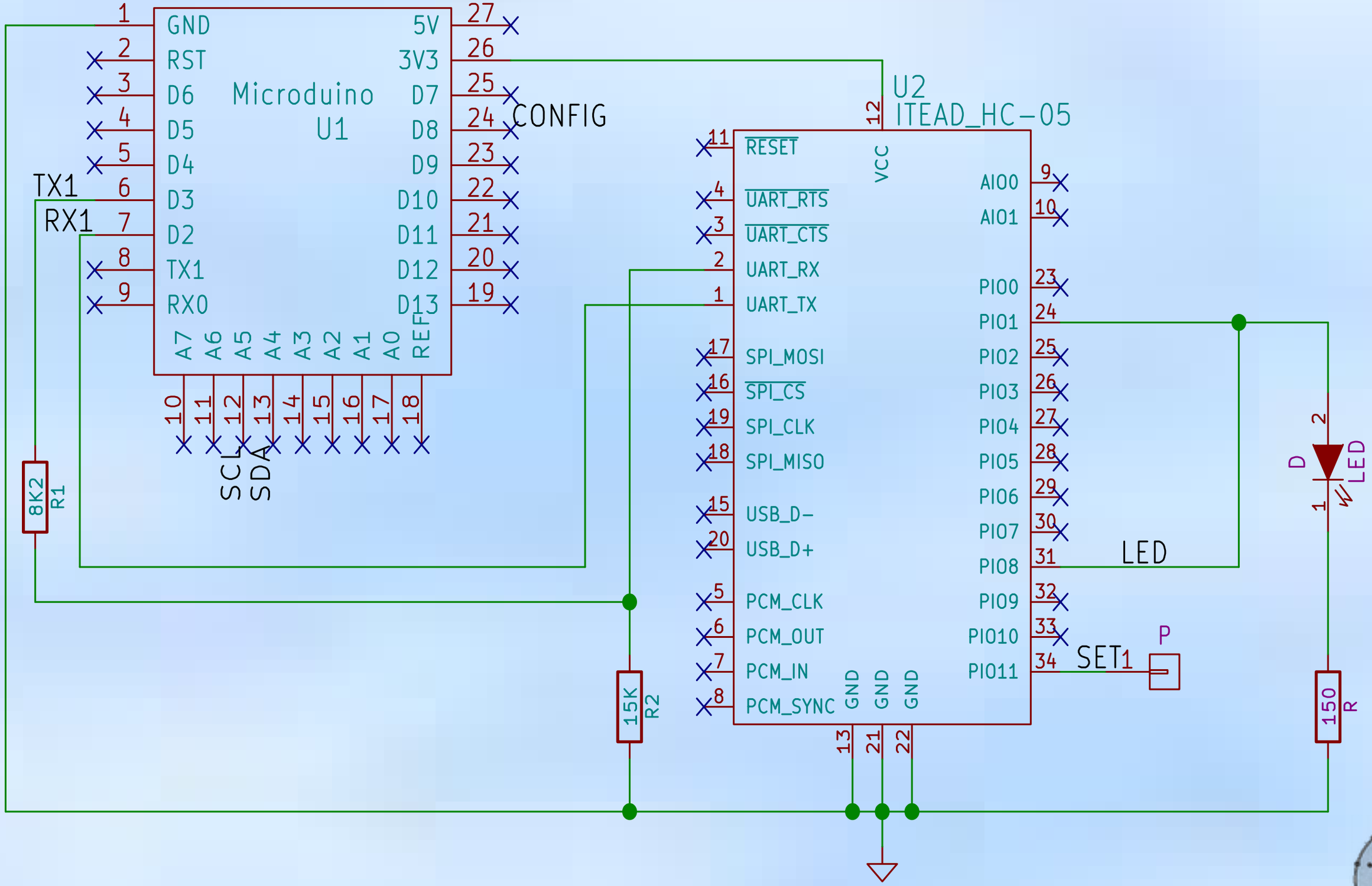

www.arduino.cc
Microduino
core+
Arduino-compatible

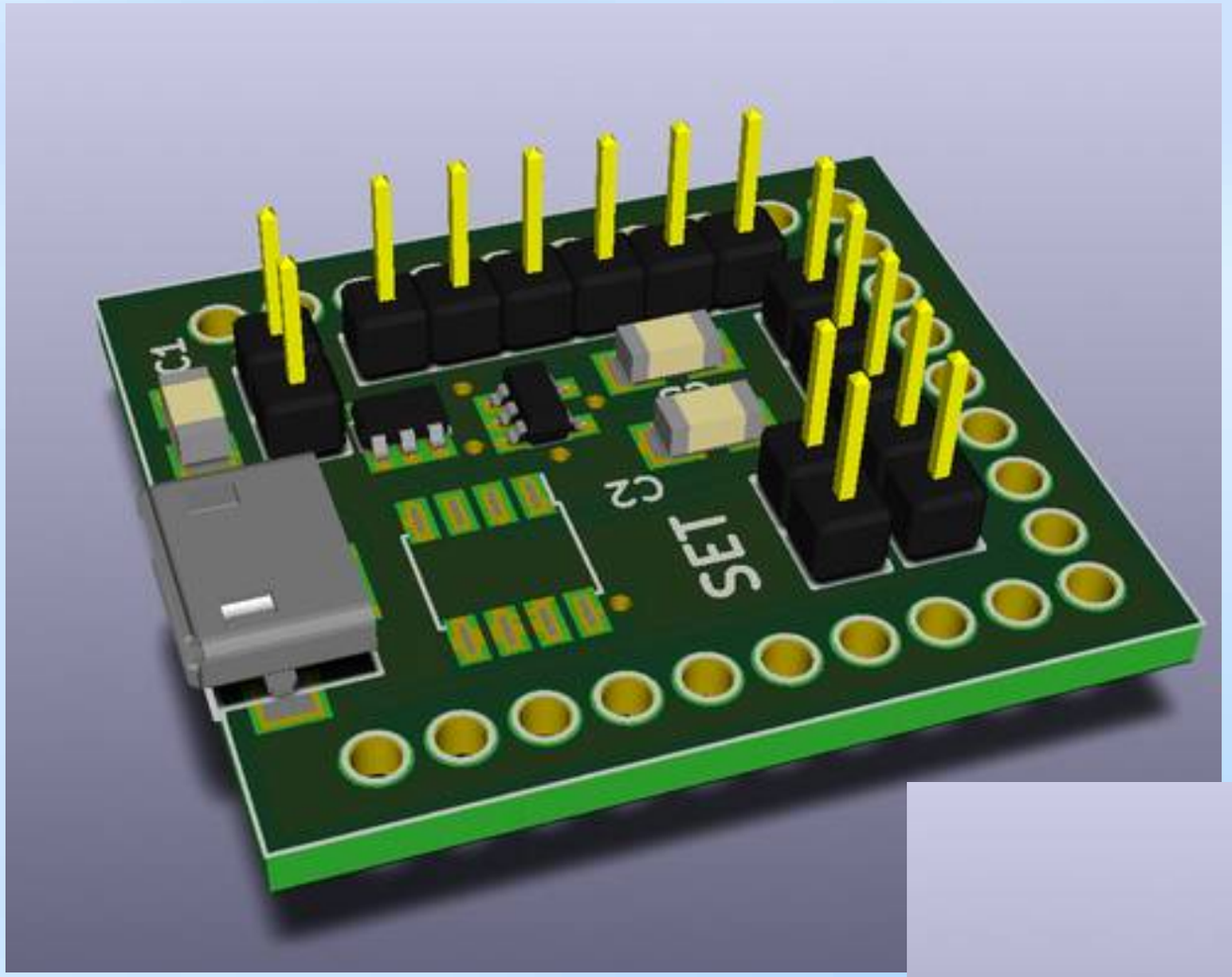


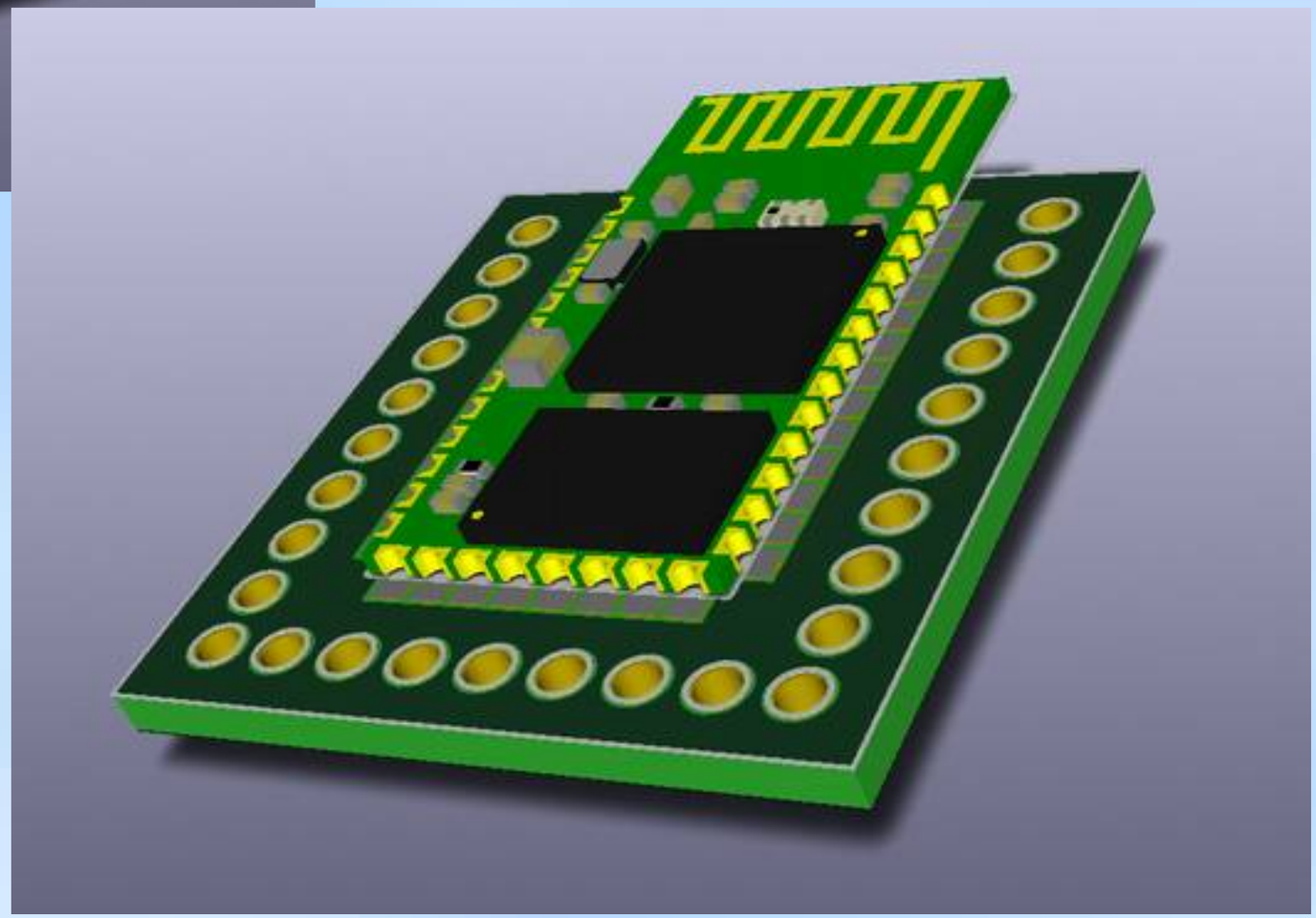
Board Stima-I2C

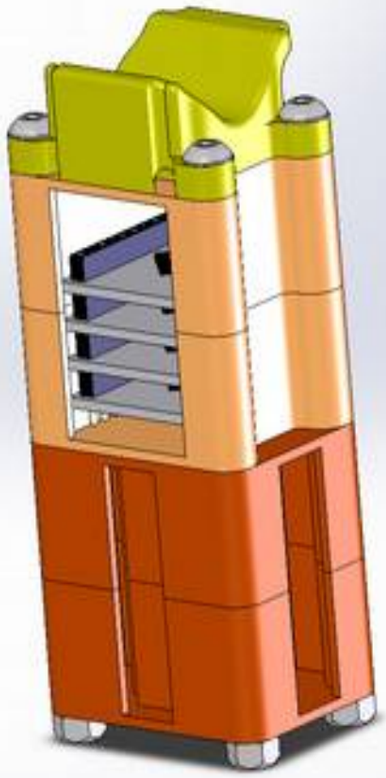


Board Stima-bluetooth




C1
C2
SET





Modulo Stima-Master

- Board microduino core+ 1284

  il componente principale e' un Atmega1284P a 16Mz ha 128K di memoria Flash, e 16K di SRAM

- Board microduino ENC

  connettore RJ45, stabilizzatore switching per il PoE e l'integrato ENC28J60

- Board STIMA-I2C

- Board microduino nRF24 (opzionale)

Modulo Stima-Satellite

Il modulo satellite è un modulo "passivo", rimane in attesa di richieste via radio da un modulo Stima-Master o STIMA-GSM/GPRS, interroga i sensori i2c e ne trasmette le misurazioni al modulo Master. E' utilizzabile solo se sul modulo Master o GSM/GPRS avete montato la board microduino nRF24. E' composto dalle seguenti schede:

- Board microduino core+ 644
- Board microduino nRF24
- Board microduino nRF24

Modulo Stima-GSM/GPRS

Questo modulo utilizza la rete di telefonia mobile. Vista la minore stabilità di questa rete di comunicazione il modulo è in grado di salvare permanentemente i dati per un lungo periodo su una SD.

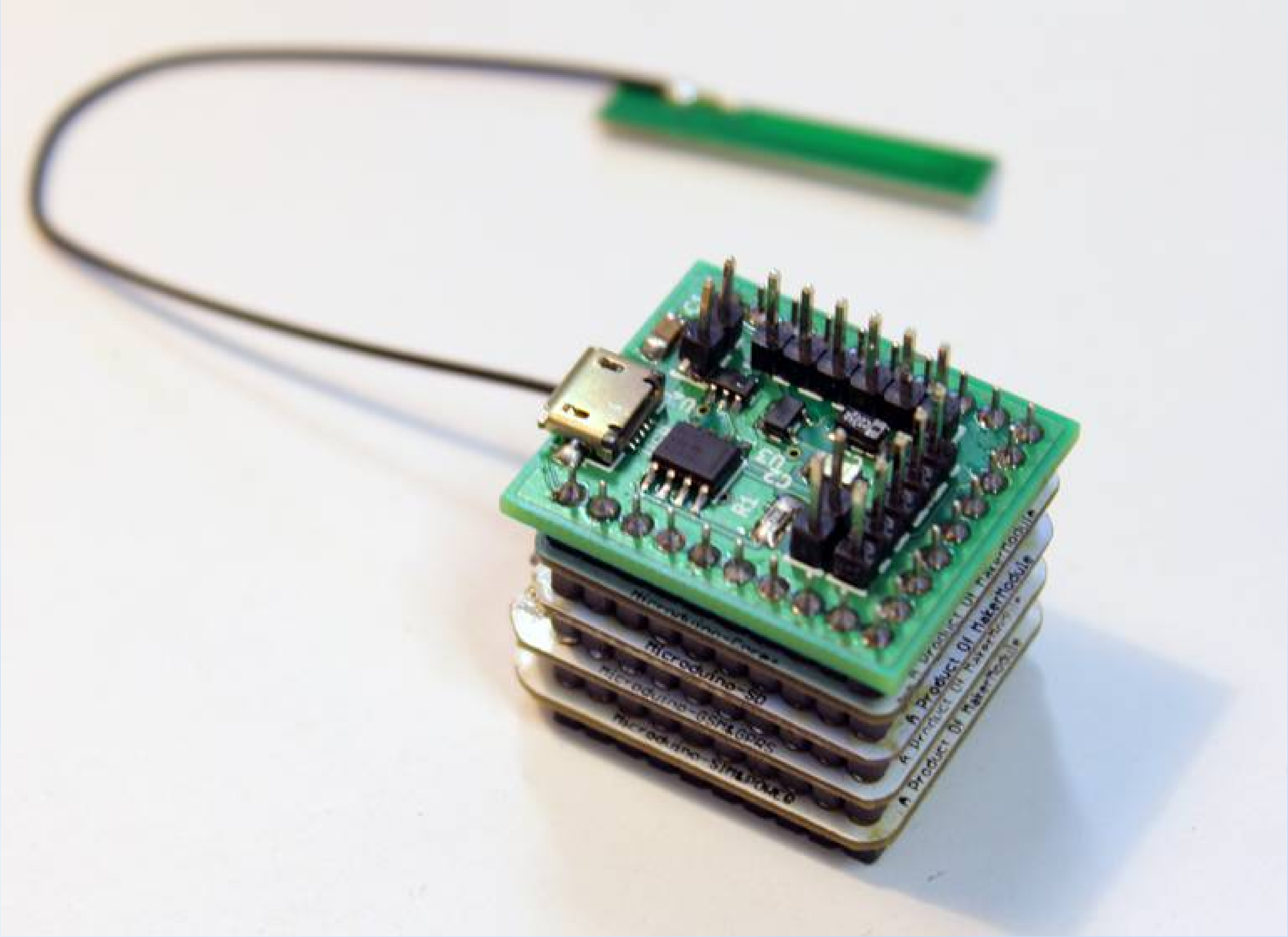
- Board microduino GPRS/GSM

Basato su sim800, un modulo quad-band GSM/GPRS,lavora con le frequenze GSM850MHz, EGSM900MHz, DCS1800MHz and PCS1900MHz. SIM800 GPRS multi-slot class 12/ class 10 e supporta gli schemi di codifica GPRS CS-1, CS-2, CS-3 e CS-4. GPRS uplink/downlink transfer rate massimi di 85.6 kbps

- Board microduino core+ 1284
- Board microduino nRF24 (opzionale)
- Board microduino SD

I modulo memorizza i dati su microsd. Quando la connessione al server viene ristabilita, i dati vengono inviati a destinazione

- Board STIMA-I2C




SIM800L
SIMCom
Microduino-SIM800H
Microduino-SIM-Shield
LED1

Modulo STIMA-I2C-GPS

Questo modulo rende disponibili i principali dati GPS su bus I2C. Questo permette di scaricare dal lavoro di monitoraggio della seriale il microcontrollore della stazione preposto all'invio dei dati. Utilizziamo la board Microduino GPS, basata su NEO 6 u-blox.

Modulo STIMA-I2C-GPS

Il firmware è in grado di gestire questi protocolli del GPS:

- NMEA
- UBX
- MEDIATEK

Il firmware è stato testato con protocollo UBX

I registri che posso essere letti sono:

- I2C_GPS_STATUS_2DFIX 2dfix achieved
- I2C_GPS_STATUS_3DFIX 3dfix achieved
- I2C_GPS_STATUS_NUMSATS Number of sats in view
- I2C_GPS_LOCATION current location 8 byte (lat, lon) int32_t
- I2C_GPS_ALTITUDE GPS altitude in meters (uint16_t)
- I2C_GPS_GROUND_SPEED GPS ground speed in m/s*100 (uint16_t)
- I2C_GPS_GROUND_COURSE GPS ground course (uint16_t)
- I2C_GPS_TIME UTC Time from GPS in hhmmss.sss * 100 (uint32_t)
- I2C_GPS_REG_YEAR Year (uint16_t) I2C_GPS_REG_MONTH Month (uint8_t)
- I2C_GPS_REG_DAY Day (uint8_t) I2C_GPS_REG_HOUR Hour (uint8_t)
- I2C_GPS_REG_MIN Minute (uint8_t) I2C_GPS_REG_SEC Second (uint8_t)

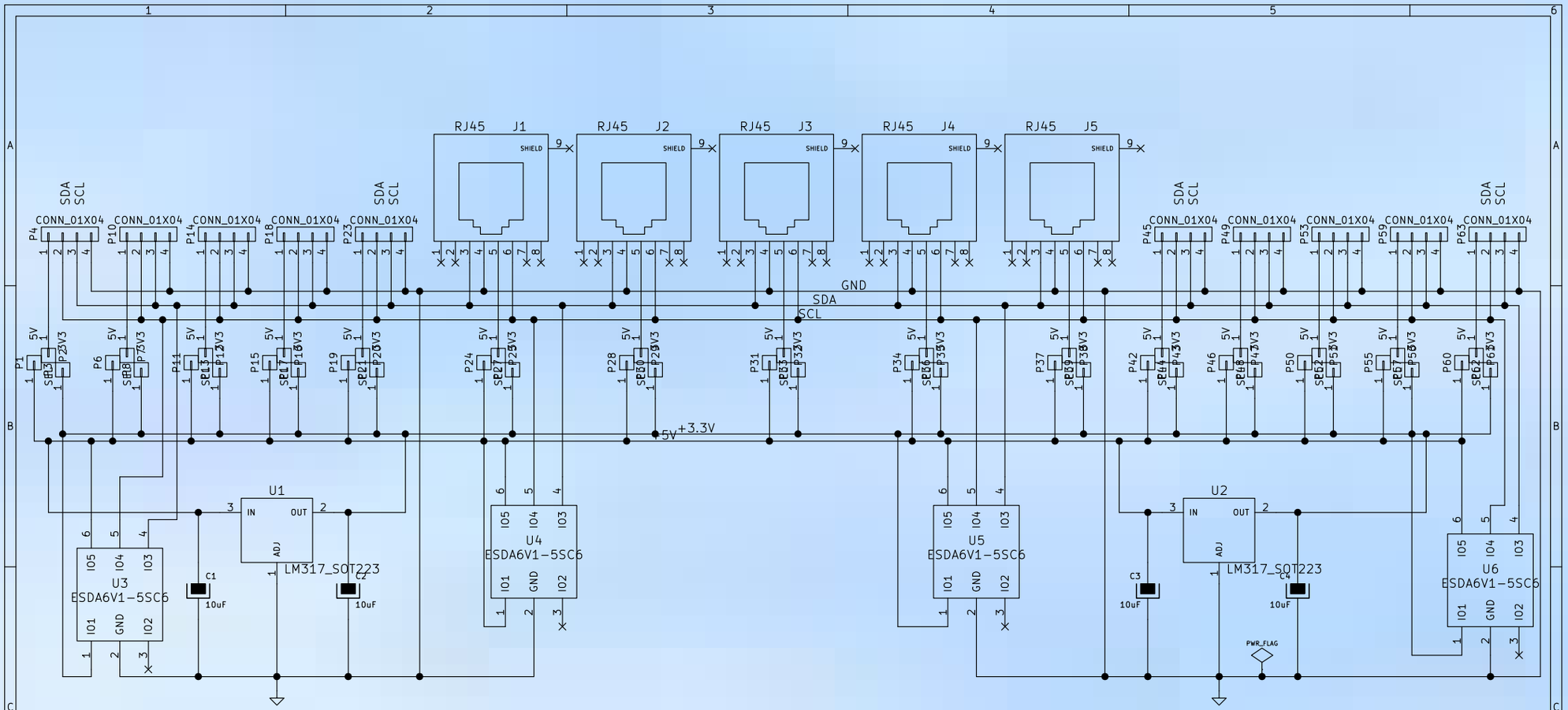
LCD display

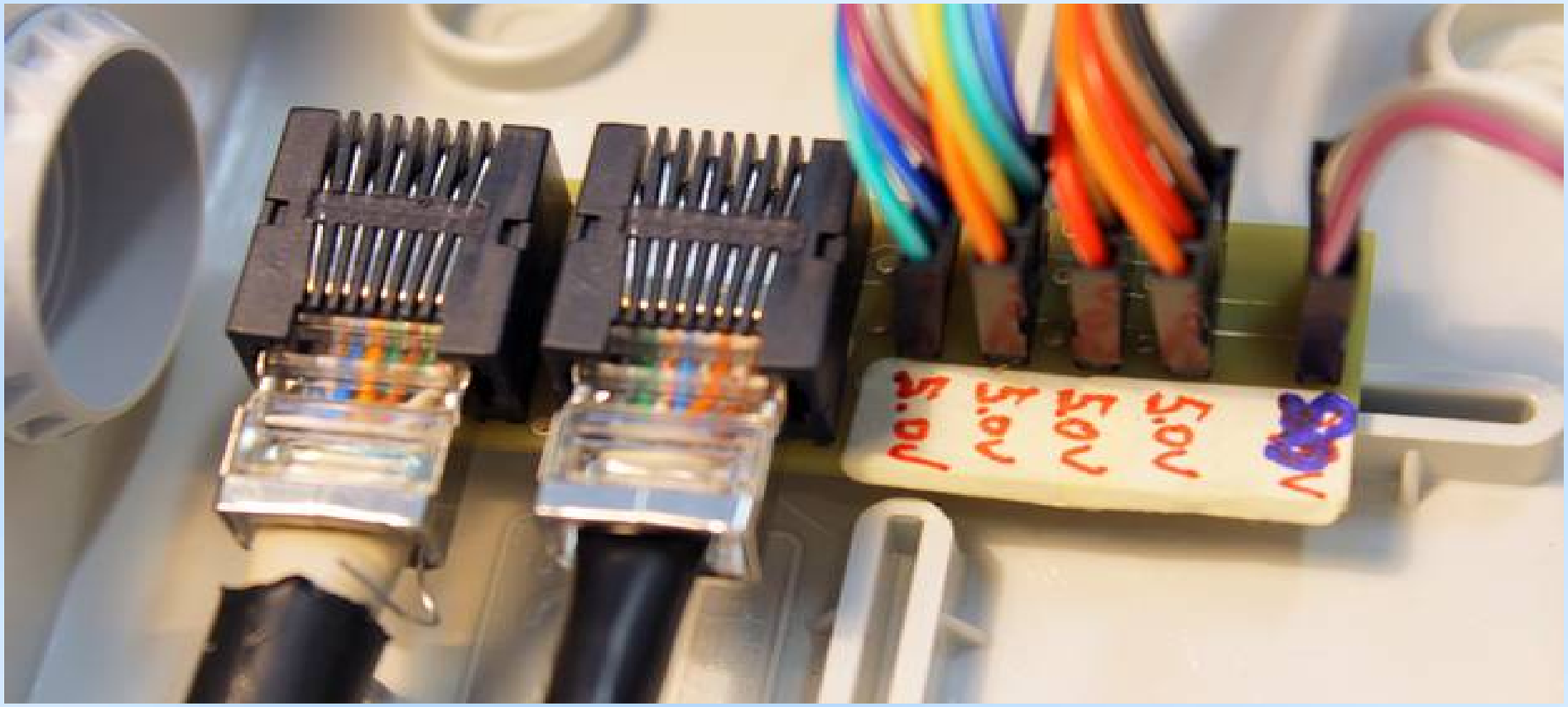
Un display Hitachi 4x20 caratteri compatibile i2c collegato ai moduli visualizza lo stato del modulo, la connessione al server, l'ora e i dati rilevati permettendo una diagnostica immediata. Può essere collegato a tutti i moduli tramite bus I2C.

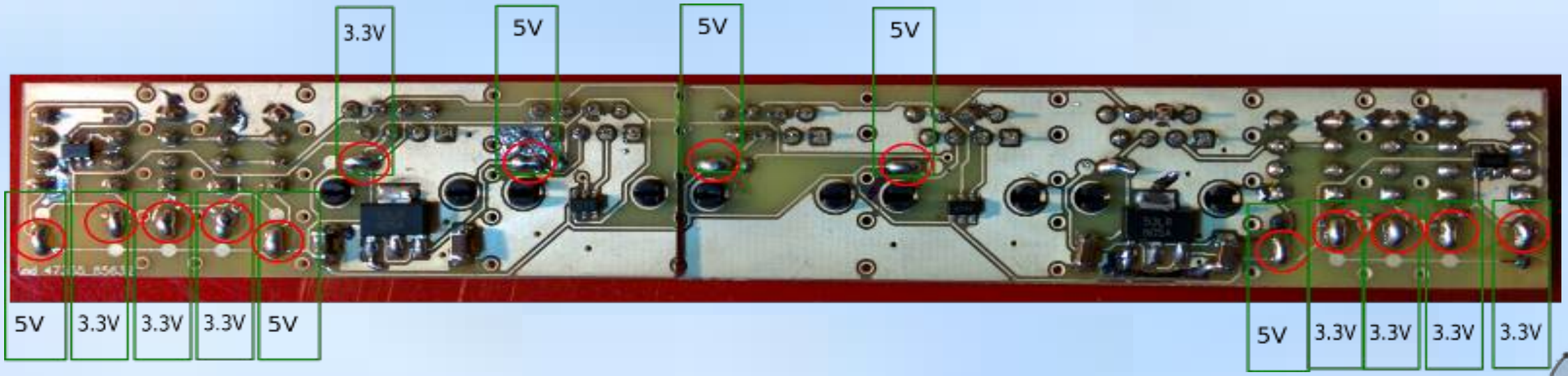
Relays

Il sistema Stima è predisposto anche per funzioni di domotica, sulla board Stima-I2C si trova un connettore pronto per collegare moduli da uno a quattro relays pilotabili con JsonRPC. Ora che abbiamo visto che questa board è utilizzata su tutti i tipi di modulo avrete capito che ogni modulo può pilotare anche attuatori creando un sistema domotico.

- La board I2C hub è in sostanza una scheda di interconnessione, non ci sono componenti intelligenti, ma solamente un paio di stabilizzatori di tensione che utilizzeremo principalmente per alimentare sensori remoti a 3.3 V.

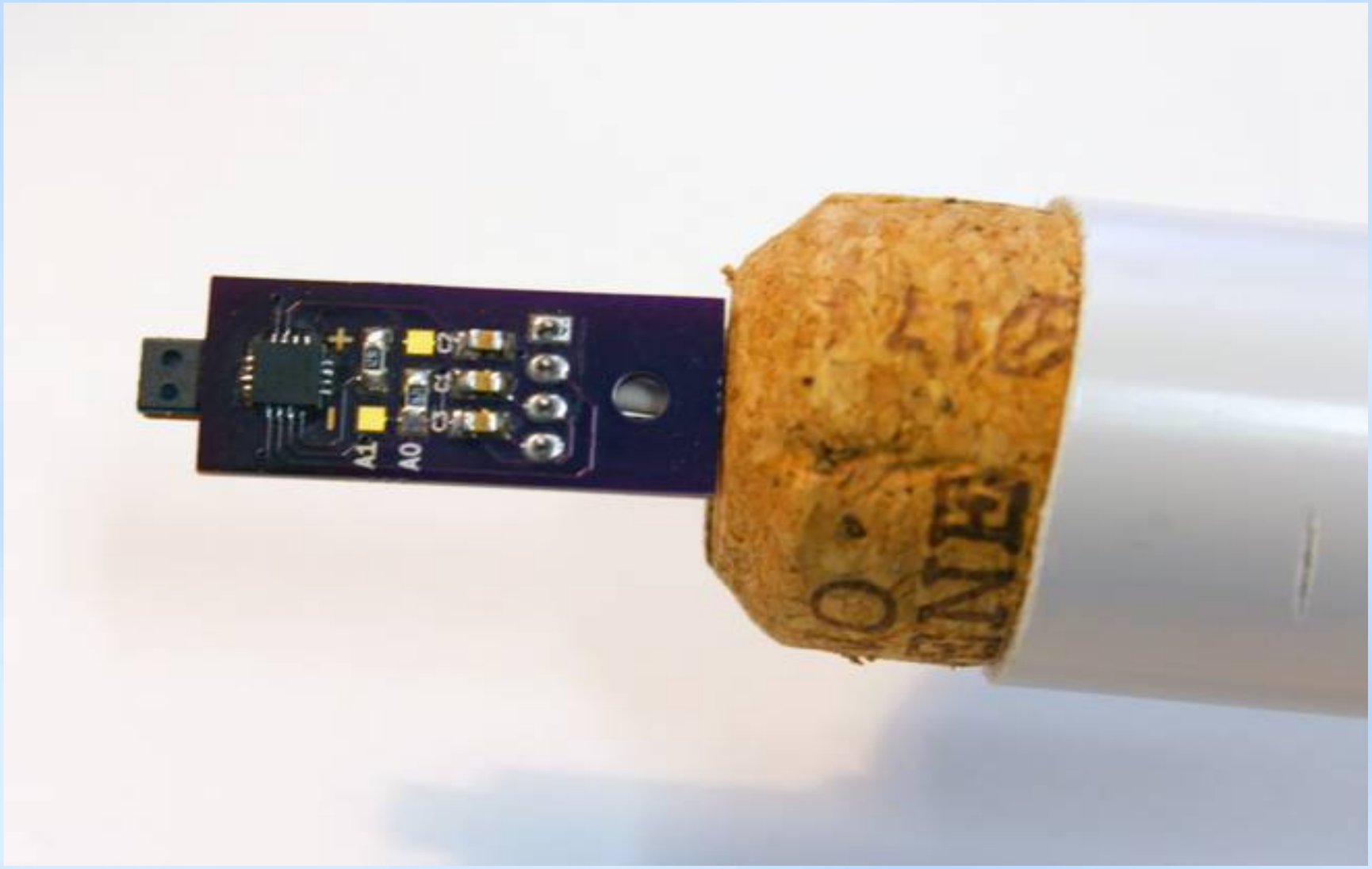





3.3V
5V
5V
5V
5V
3.3V
3.3V
3.3V
5V
5V
3.3V
3.3V
3.3V
3.3V

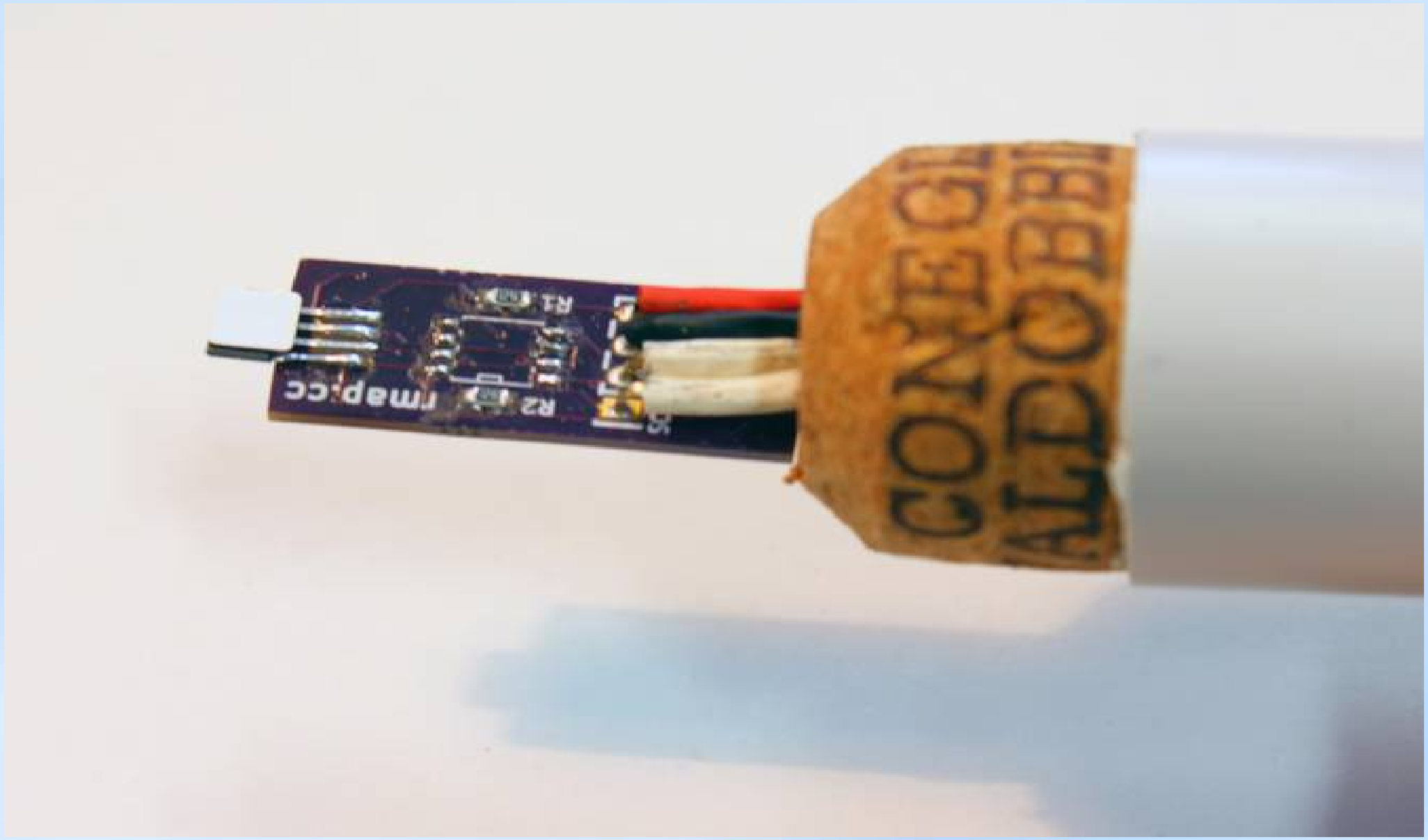
Sensori temperatura

- ADT7420 è il sensore di temperatura compatibile col bus I2C con la maggiore accuratezza disponibile al momento, il datasheet riporta −10°C to +85°C ±0.2°C, bit resolution: 0.0078°C, Ultra low temperature drift: 0.0073°C



Sensore umidità

- Sensore HIH6100 dedicato alla misura di umidità, ±4.0 %RH accuratezza con filtro idrofobico opzionale



Sensore di pressione

- BMP180 della Bosch. Pressure range: 300/1100hPa (+9000m/-500m sul livello del mare); Accuratezza relativa tipica 950/1050 hPa 25 °C : +/- 0.12 hPa; Accuratezza relativa tipica 700/900 hPa 25/40 °C : +/- 0.12 hPa. Il BMP180 è basato sulla tecnologia piezo-resistiva, calibrato e compensato in temperatura.

Modulo STIMA-I2C-wind

Il firmware ha due modalità di funzionamento: la prima chiamata one-shot che vede l'effettuazione di un campionamento una tantum e l'altra che continua ad effettuare misure ed elaborare osservazioni in modo ciclico.

È possibile impartire alcuni comandi:

– I2C_WIND_COMMAND_ONESHOT_START

– I2C_WIND_COMMAND_ONESHOT_STOP

tra uno start e uno stop bisogna attendere circa 3 secondi (dipendente dalla strumentazione). Dopo uno stop è possibile leggere questi registri:

- I2C_WIND_DD direzione vento
- I2C_WIND_FF forza del vento
- I2C_WIND_U componente u
- I2C_WIND_V componente v

Modulo STIMA-I2C-wind

Con la modalità ciclica dopo il comando I2C_WIND_COMMAND_STOP e si potranno leggere direttamente i registri che riporteranno dati appena elaborati; oltre ai precedenti sarà possibile leggere anche questi registri:

- I2C_WIND_MEANU media U 10 minuti
- I2C_WIND_MEANV media V 10 minuti
- I2C_WIND_PEAKGUSTU picco raffica U 10 minuti
- I2C_WIND_PEAKGUSTV picco raffica V 10 minuti
- I2C_WIND_LONGGUSTU lunga (60s) raffica U 10 minuti
- I2C_WIND_LONGGUSTV lunga (60s) raffica V 10 minuti
- I2C_WIND_MEANFF media forza del vento 10 minuti
- I2C_WIND_SIGMA deviazione standard forza 10 minuti
- I2C_WIND_SECTOR(1...8) frequenza nel settore 1...8 (sector 1 da -22.5 to +22.5)
- I2C_WIND_SECTORCALM frequenza calma di vento

Modulo STIMA-I2C-wind

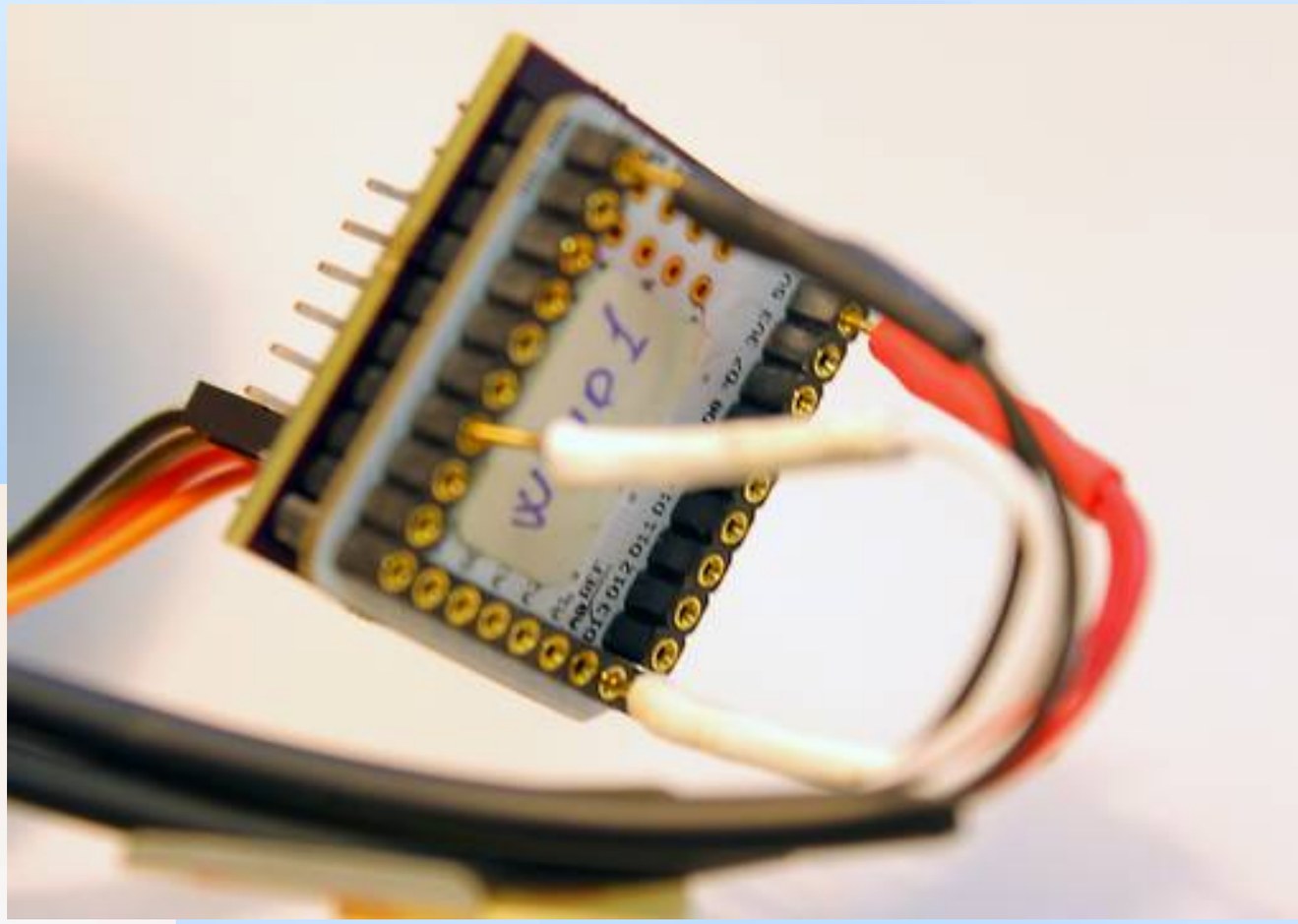
- Per la lettura dei registri viene utilizzata la tecnica del double buffering per rendere le letture dei registri atomiche; a ogni comando stop i buffer vengono scambiati.

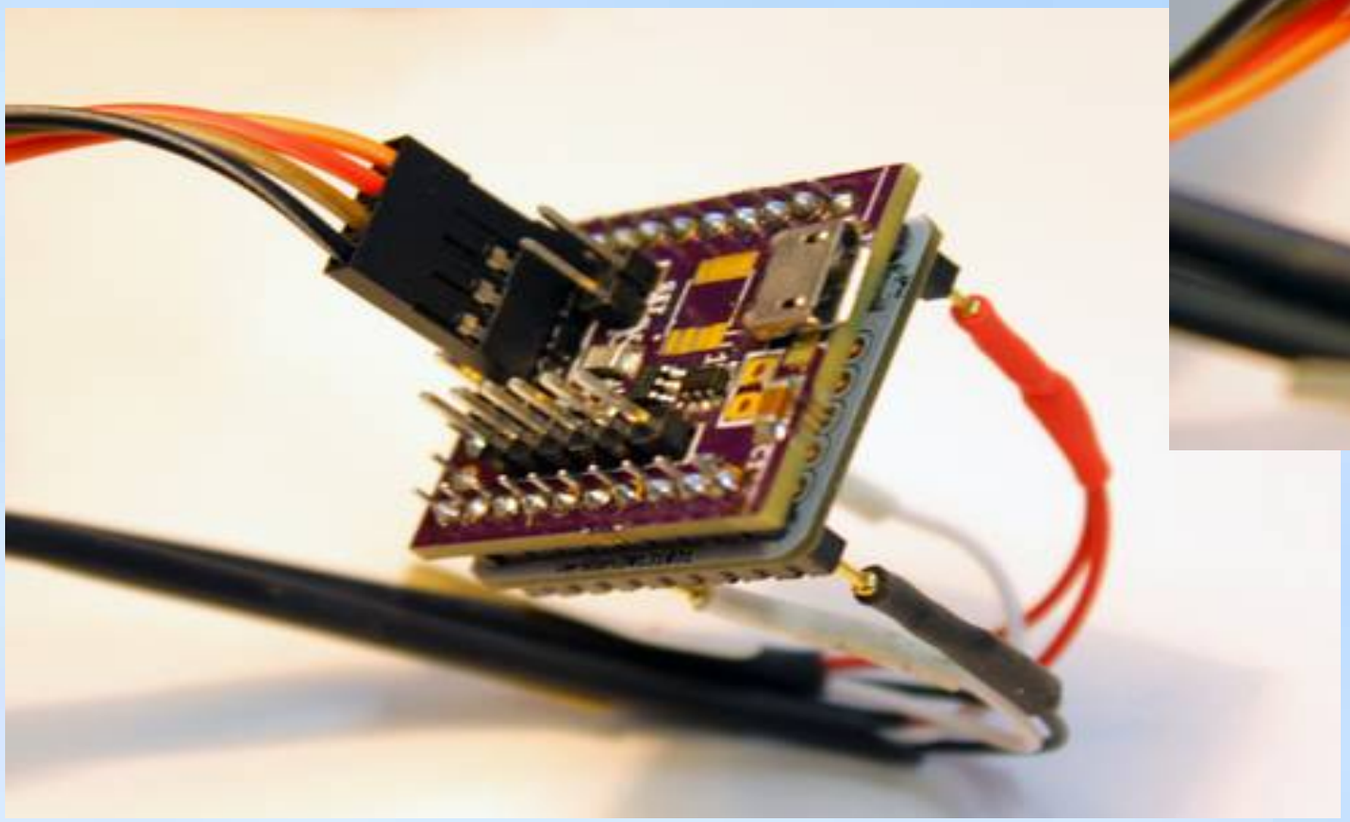
- Per l'elaborazione dei dati invece vengono usati dei circular buffer che permettono di elaborare in continuo medie, massime ... Per risparmiare memoria sono usati due livelli di circular buffer a risoluzioni temporali differenti e due livelli di elaborazione differenti che sfruttano la proprietà distributiva degli algoritmi

Modulo STIMA-I2C-wind

Il modulo si realizza impilando le seguenti board:

- board microduino core
- board STIMA-I2C





Anenometro Davis

Strumento integrato con potenziometro 20KΩ per direzione e un contatto reed per intensità.

	Range	Accuracy	Resolution	Measurement Timing
Wind Speed	0.5 to 89 m/s	± 1 m/s or ±5%, whichever is greater	0.1 m/s	Sample Period 2.25 seconds
Wind Direction	0° to 360°	±7°	1°	Filter Time Constant (typical) 8 seconds

Anemometro Inspeed

Inspeed VORTEX Series II HEAVY DUTY ANEMOMETER, E-VANE2, con sensori a effetto di Hall. Il sensore Hall richiede una alimentazione a 5VDC. L'anemometro inspeed offre alcuni vantaggi rispetto a quello Davis dovuti ai sensori ad effetto di Hall che eliminano la necessità di sistemi antirimbalzo, riducono gli atriti e hanno zero deadband. Caratteristiche E-VANE2:

- Supply voltage 2.7 to 5.5 VDC
- Current 12 mA typical
- Output 5% to 95% of input voltage
- Output impedance: 500 Ohms
- Input impedance: min. 50 kOhms

Anemometro Inspeed

	Sensor type	Range	Accuracy	Resolution
Wind Speed	3-Cup Lexan rotor Hall Sensor	64 m/s	+/-4% of reading or 0.5 m/s, whichever is greater	
Wind Direction	Balanced wind vane connected to an active, non-contact, zero friction Hall Effect sensor	Full 360 degrees , zero deadband	+/- 1.7% of Full Scale (+/- 6 degrees)	0.025 degrees

Modulo STIMA-I2C-rain

- Questo modulo funziona connesso a pluviometri a doppia vaschetta basculante
- Il firmware ha per ora una sola modalità di funzionamento ossia quella one-shot
- Possibili comandi
  - START : reset del contatore di basculate (da effettuare sempre all'avvio)
  - STOP : prepara alla lettura del contatore rendendo disponibile l'ultimo stato nel buffer di lettura
  - STARTSTOP : effettua uno START e uno STOP in modo atomico
- In questo modo il valore cumulato è sempre garantito particolrmente se verrà prevista la presenza di una batteria tampone.
- Dopo uno stop è possibile leggere questi registri:
  - I2C_RAIN_TIPS 0x01 Numero basculate

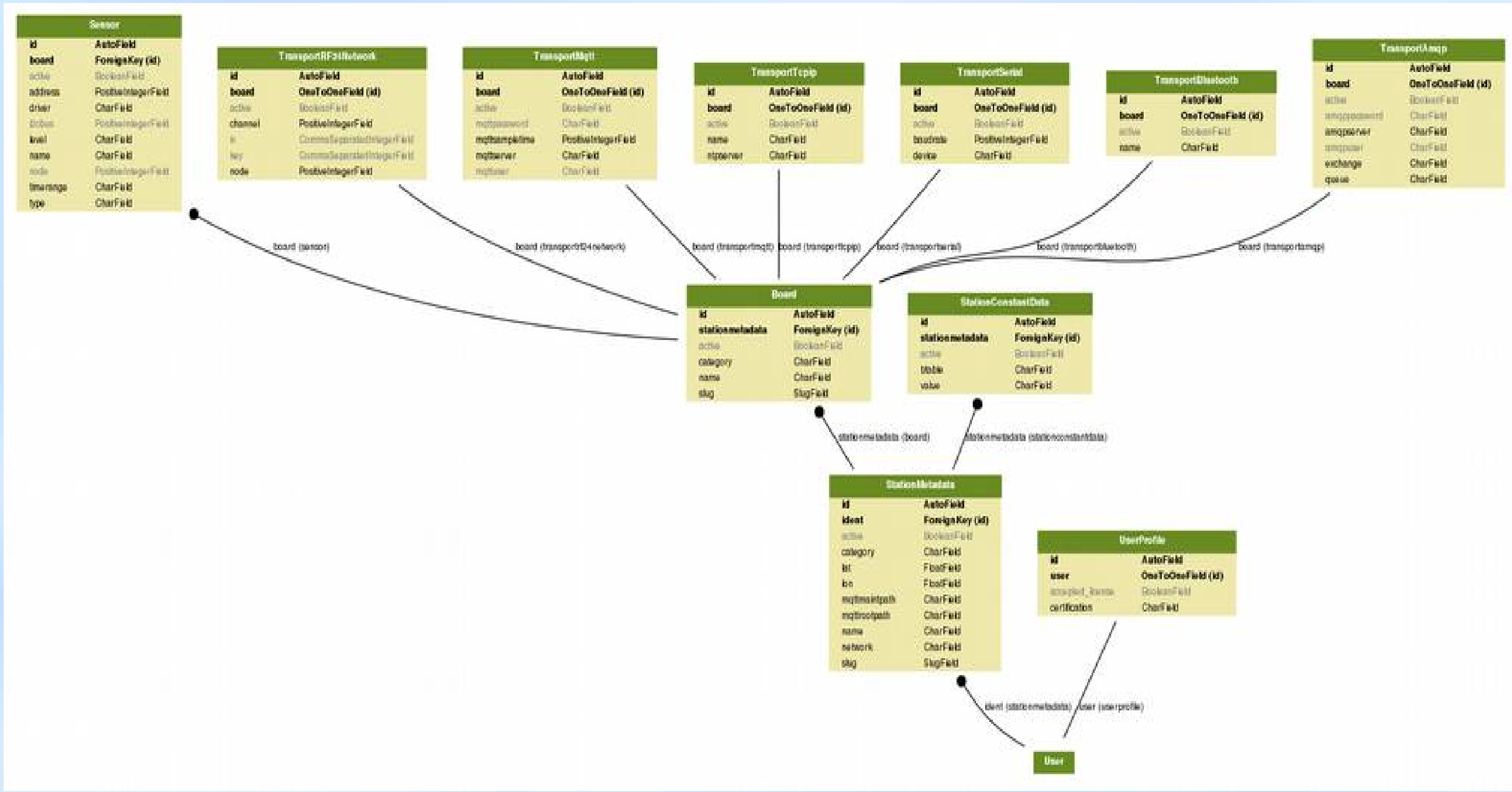
Modulo STIMA-I2C-rain

Il modulo si realizza impilando le seguenti board:

- board microduino core
- board STIMA-I2C

Collegabile al pluviometro Davis

Tabelle e relazioni dei metadati di stazione



L'applicazione RMAP

- Questo programma interagisce con l'utente tramite interfaccia grafica e permette la pubblicazione di dati ambientali sul server rmap.cc. I dati possono essere rilevati manualmente e a vista oppure con i moduli STIMA

  L'APP guida attraverso una sequenza di fasi:

- identificazione della propria posizione
- inserimento dei dati manuali
- connessione e attivazione del dispositivi Stima per il rilevamento automatico dei dati
- connessione e invio dei dati al server

  L'APP ha due modalità di funzionamento: una interattiva e una in background.

Tecnologie utilizzate nell'applicazione RMAP

- **Django**

  Django è un web framework per lo sviluppo di applicazioni web, scritto in linguaggio Python. Fornisce funzionalità che facilitano lo sviluppo rapido di applicazioni per la gestione di contenuti.

  Principali funzionalità:

  - Astrazione del **database** relazionale ad oggetti
  - Sistema di **template** basato su tag con ereditarietà dei template
  - Supporto per **localizzazione**
  - Sistema di gestione degli **utenti** e loro **autenticazione** nell'applicazione Web
  - Sistema per la creazione e la validazione di **form** HTML

Kivy

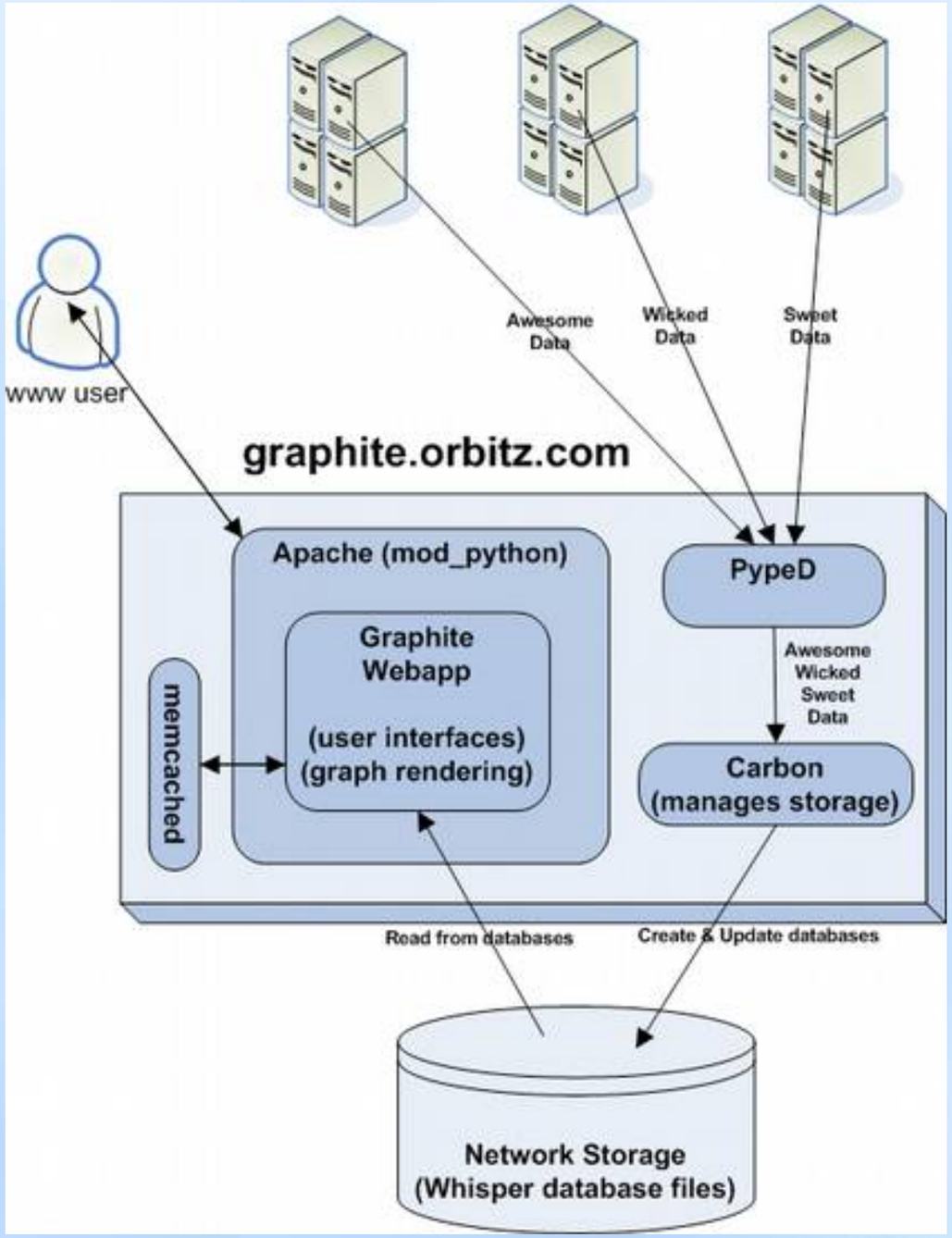
Kivy è una libreia Python per lo sviluppo di applicazioni su device mobili e/o multi-touch con una interfaccia utente naturale (NUI). Funziona su Android, iOS, Linux, OS X, e Windows. Distribuito con la licenza MIT, Kivy è software libero.

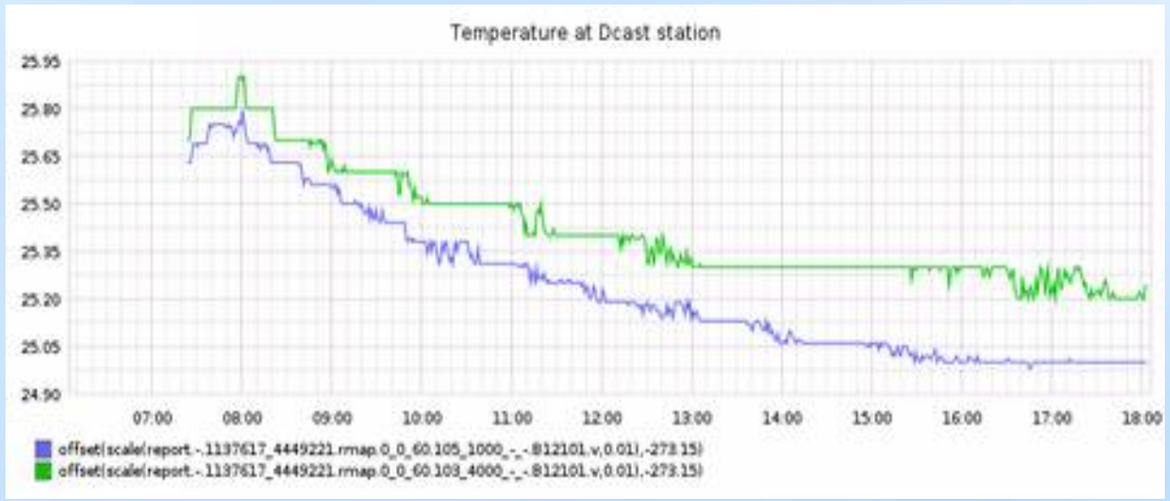
Cellulari e tablet hanno portato con sé un cambiamento drammatico nell'uso delle applicazioni. La compatibilità è diventata essenziale e ha aumentato il tipo di interazione che gli utenti si aspettano : gesti , multi-touch , animazioni e penne magiche. Kivy è una soluzione open source Python che copre queste esigenze di mercato con un approccio di sviluppo facile da imparare e rapida . Kivy sta crescendo rapidamente e guadagnando attenzione come alternativa alle piattaforme di sviluppo standard.

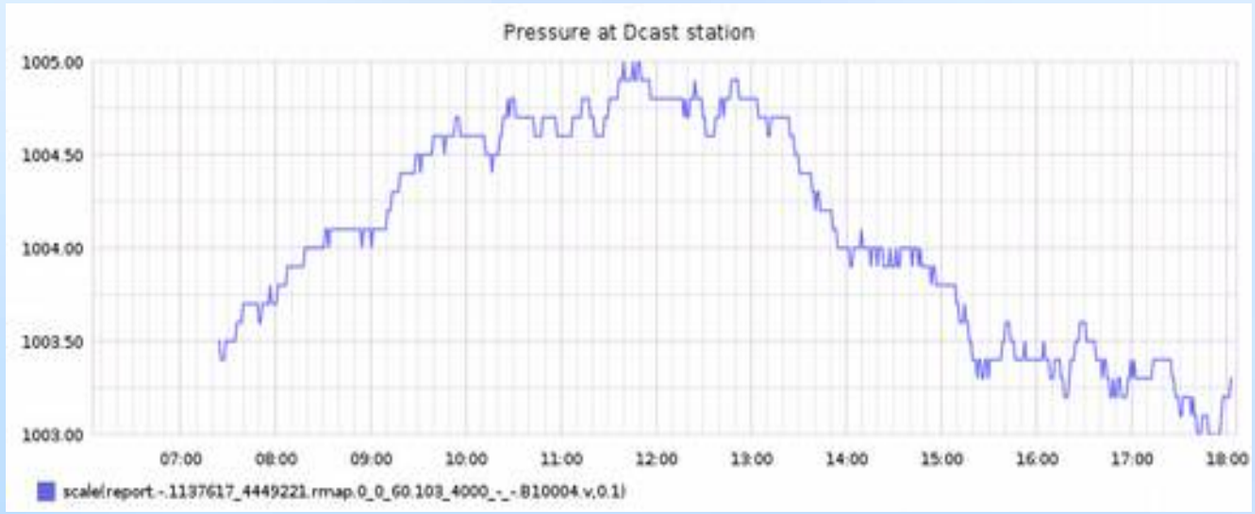
Graphite - Scalable Realtime Graphing

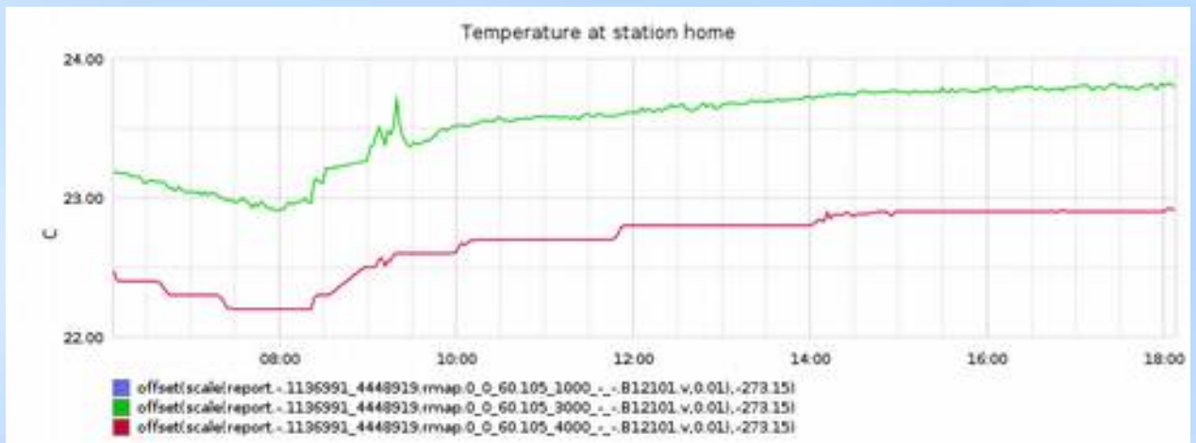
What is Graphite?

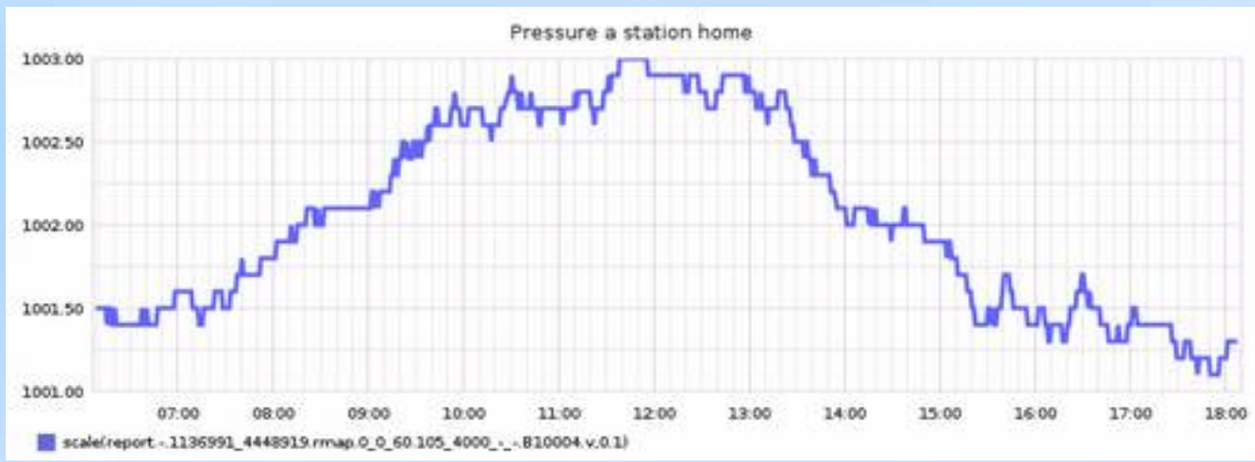
Graphite is a highly scalable real-time graphing system. As a user, you write an application that collects numeric time-series data that you are interested in graphing, and send it to Graphite's processing backend, carbon, which stores the data in Graphite's specialized database. The data can then be visualized through graphite's web interfaces.

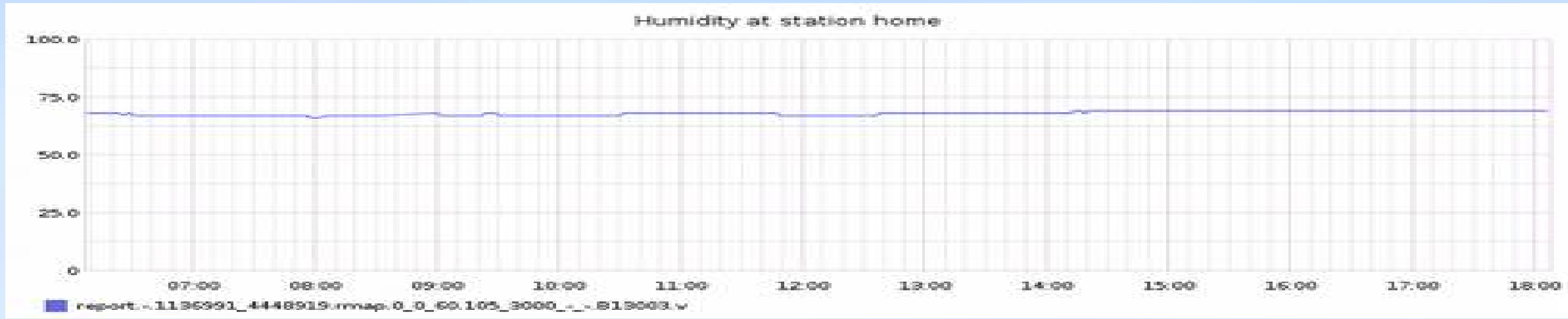



Temperature at Dcast station
25.95
25.80
25.65
25.50
25.35
25.20
25.05
24.90
07:00
08:00
09:00
10:00
11:00
12:00
13:00
14:00
15:00
16:00
17:00
18:00
offset(scale(report.-.1137617_4449221.rmap.0_0_60.105_1000_-_-.B12101.v,0.01),-273.15)
offset(scale(report.-.1137617_4449221.rmap.0_0_60.103_4000_-_-.B12101.v,0.01),-273.15)


Pressure at Dcast station
1005.00
1004.50
1004.00
1003.50
1003.00
07:00
08:00
09:00
10:00
11:00
12:00
13:00
14:00
15:00
16:00
17:00
18:00
scale(report.-.1137617_4449221.rmap.0_0_60.103_4000_-_-.B10004.v,0.1)

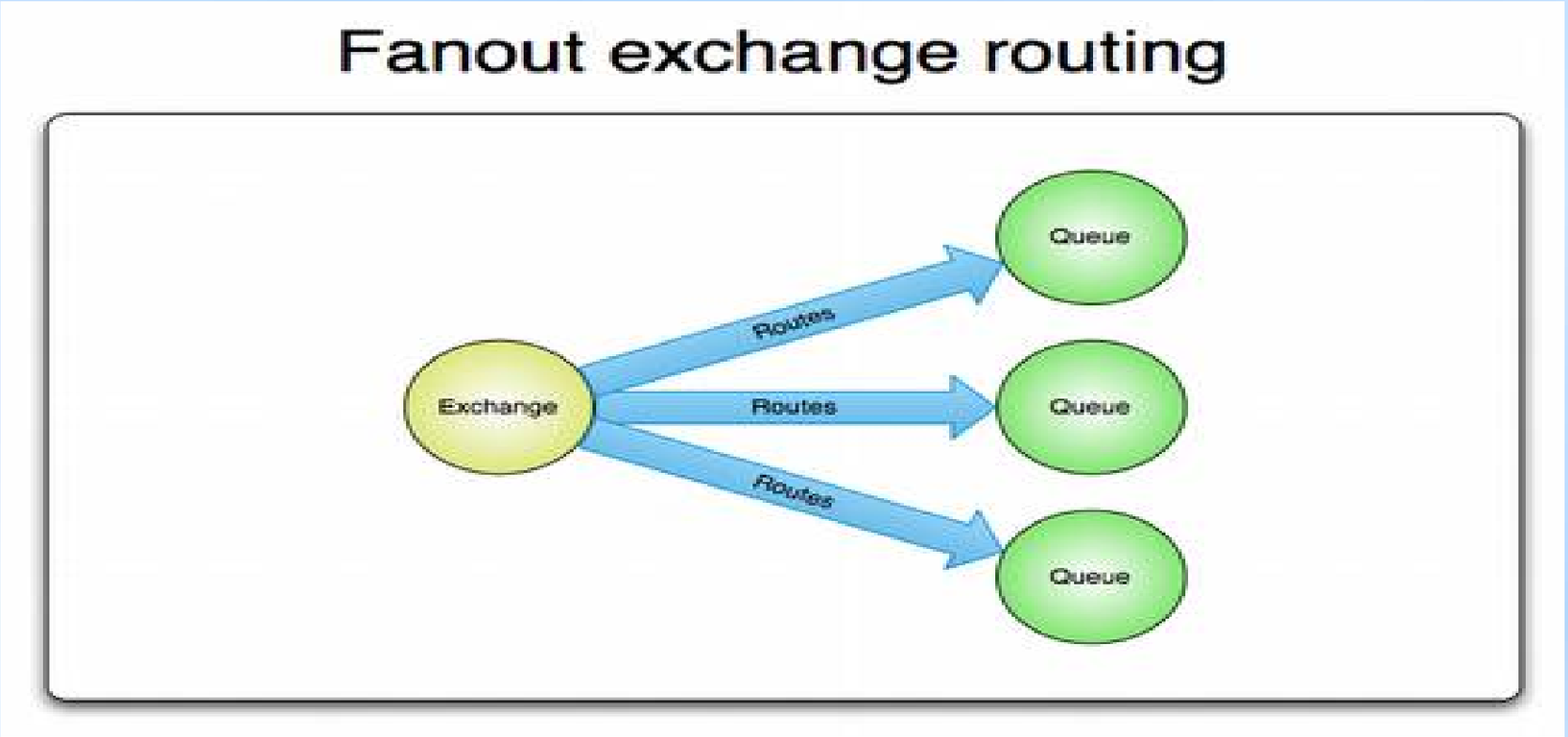

Temperature at station home
24.00
23.00
22.00
C
08:00
10:00
12:00
14:00
16:00
18:00
offset(scale(report.-.1136991_4448919.rmap.0_0_60.105_1000_-_-.B12101.v,0.01),-273.15)
offset(scale(report.-.1136991_4448919.rmap.0_0_60.105_3000_-_-.B12101.v,0.01),-273.15)
offset(scale(report.-.1136991_4448919.rmap.0_0_60.105_4000_-_-.B12101.v,0.01),-273.15)


Pressure a station home
1003.00
1002.50
1002.00
1001.50
1001.00
07:00
08:00
09:00
10:00
11:00
12:00
13:00
14:00
15:00
16:00
17:00
18:00
scale(report.-.1136991_4448919.rmap.0_0_60.105_4000_-_-.B10004.v,0.1)


Humidity at station home
100.0
75.0
50.0
25.0
0
07:00
08:00
09:00
10:00
11:00
12:00
13:00
14:00
15:00
16:00
17:00
18:00
report.-.1136991_4448919.rmap.0_0_60.105_3000_-_-.B13003.v

Rabbitmq

- message broker software that implements the Advanced Message Queuing Protocol (**AMQP**).
- **Exchanges** are AMQP entities where messages are sent.
- **Queue** store messages that are consumed by applications
- A **fanout exchange** routes messages to all of the queues that are bound to it



Upload messages

- **Shovel plugin**
  - The high level goal of a shovel is to reliably and continually take messages from a queue (a source) in one broker and publish them to exchanges in another broker (a destination).
  - The primary advantages of a shovel are:
    - Loose coupling

      A shovel can move messages between brokers (or clusters) in different administrative domains:
      - they may have different users and virtual hosts;
      - they may run on different versions of RabbitMQ and Erlang.
    - WAN-friendly
      - The Shovel plugin uses AMQP to communicate between brokers, and is designed to tolerate intermittent connectivity without message loss.
    - Highly tailorable
      - When a shovel connects (either to the source or the destination) it can be configured to perform any number of explicit methods. For example, the source queue need not exist initially, and can be declared on connect.

DB-All.e

- tool per gestione dati puntuali meteorologici basato sulla loro rappresentazione fisica
- sviluppato dal SIMC, utilizzato per verifica modelli, applicazione operativa filtro di Kalman, casi studio
- Corredato di Provami, sofisticato programma interattivo per la:
  - Visualizzazione
  - Ricerca
  - Modifica
  - Esportazione

DB-All.e

DB-All.e is a fast on-disk database where meteorological observed and forecast data can be stored, searched, retrieved and updated. Many modern meteorological applications, like data assimilation, quality control and verification, need complex procedures for input of meteorological data, such as decoding and standardisation, and their organisation in memory. The need to manage a high number of measurement points and the need of long spans of space and time dimensions lead to a disproportionate use of RAM and increase the complexity of programming.

This framework allows to manage large amounts of data using its simple Application Program Interface (API), and provides tools to visualise, import and export in the standard formats BUFR and CREX.

DB-all.e

- **Fortran (77,2003)**, **C++** and **Python** API are provided.
- Developed using **ODBC** and optimized DB dependant programming layer
- To make computation easier, data is stored as **physical** quantities, that is, as values of a variable in a specific point of space and time, rather than as a sequence of encoded reports.
- Representation is in 7 dimensions: **observation network**, **x**, **y**, **z**, **datetime**, **timerange**, **variable**, where x,y are geographic coordinate, z table driven vertical coordinate, datetime the reference time, timerange table driven observation and forecast specification, variable table driven unique definition.
- Any data may have **attribute**, containing more information linked to the data.
- Unlimited station information data are possible.
- **Real**, **integer** and **character** data type are supported.
- It is **fast** for both read and write access.

DB-all.e

- It is based on physical principles, that is, the data it contains are defined in terms of **homogeneous** and **consistent** physical data. For example, it is impossible for two incompatible values to exist in the same point in space and time.
- It can manage **fixed stations** and **moving stations** such as **airplanes** or **ships**.
- It can manage both **observational** and **forecast** data.
- It can manage data along **all three dimensions in space**, such as data from soundings and airplanes.
- Report information is preserved. It can work based on physical parameters or on report types.

The Arkimet archiving system

- Set di tool per archiviazione e distribuzione di dati ambientali
- Accesso locale (filesystem), remoto (HTTP) omogeneo tramite CLI
- Integrità dei dati: sono trattati come una stringa binaria opaca, in sola lettura e mai modificata
- E' possibile estendere facilmente i formati supportati (attualmente GRIB, BUFR, ODIMH5)
- Deploy molto semplice e veloce
- Arkiweb: interfaccia web http://www.smr.arpa.emr.it/arkiweb
- Sviluppato dal SIMC Licenza GPLv2+
- http://svn.smr.arpa.emr.it/arkimet/arkimet/trunk

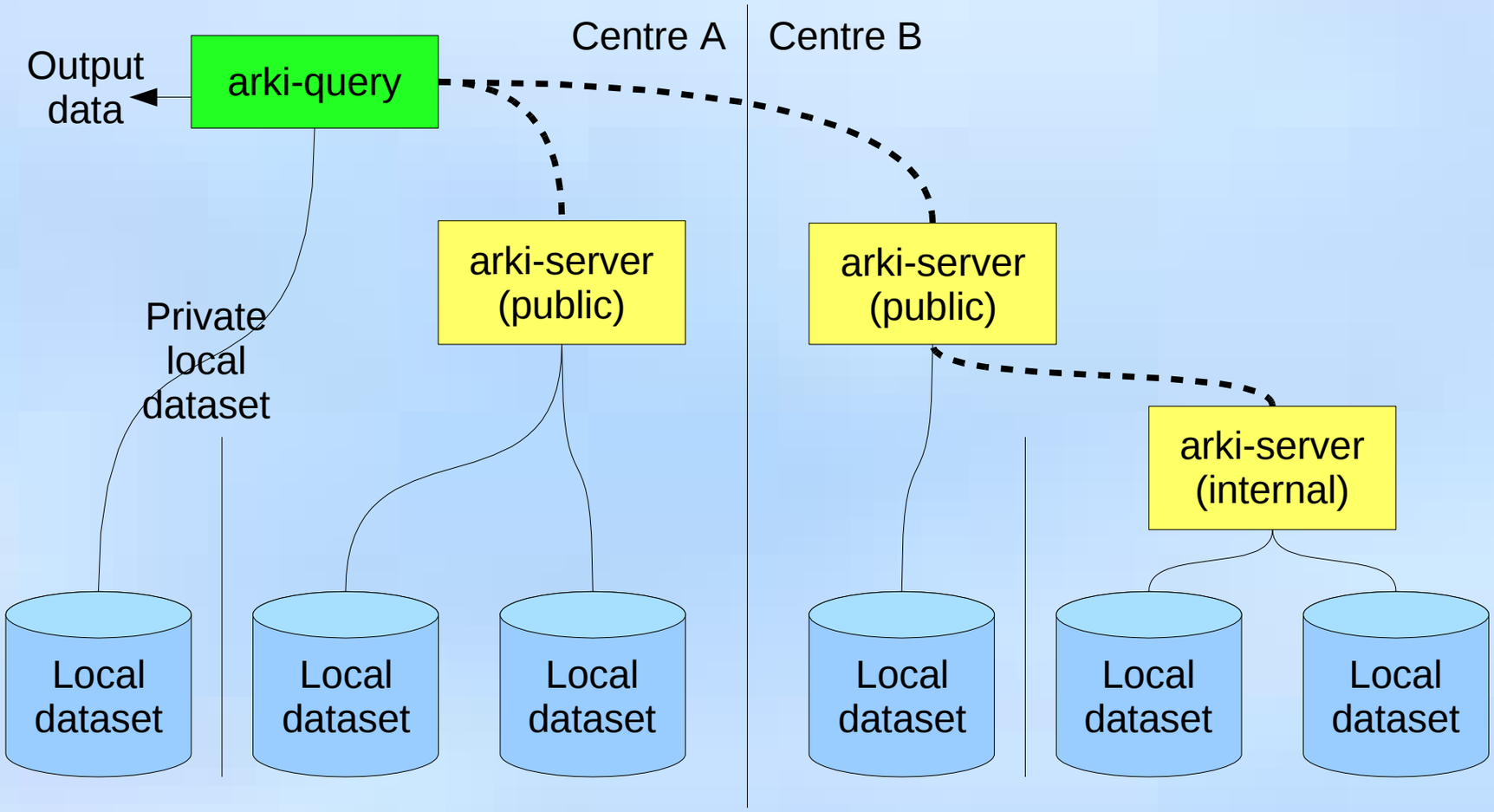
Arkimet: come lavora

- Data is examined and metadata are extracted
- Data and metadata are acquired into datasets
- Datasets are self-contained collections of homogeneous information
- Datasets store data, metadata, and also summaries of the data
- A summary can be used to explore the contents of a dataset, or the output of a query, without extracting the data

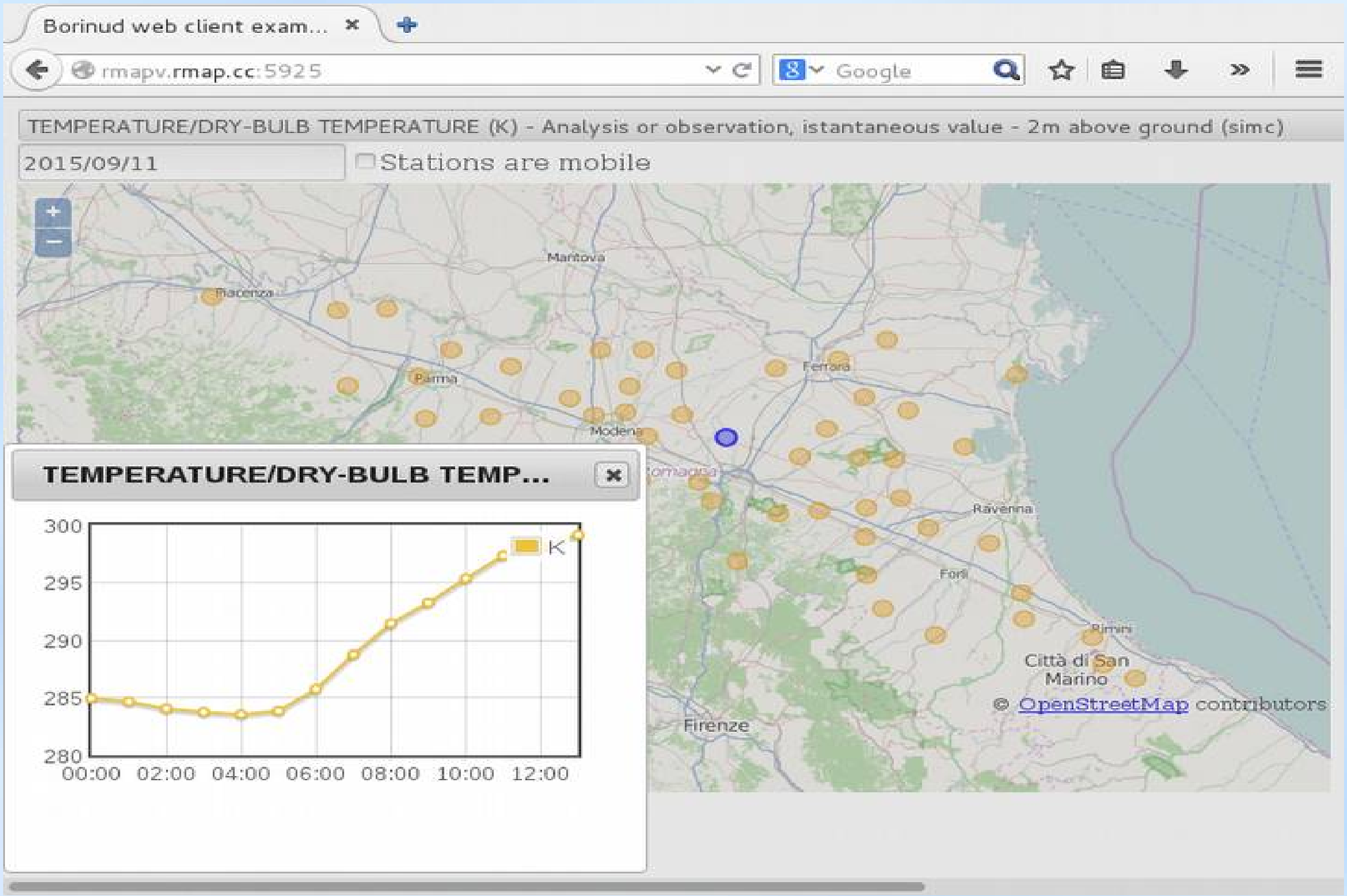
Arkimet: distributed

- Data is accessible locally and remotely, in the same way
- Remote access uses a client-server model, over standard HTTP
- Any centre / unit can deploy their own
- Server can share local and remote datasets
  - It is possible to create a public front-end server that aggregates several internal servers

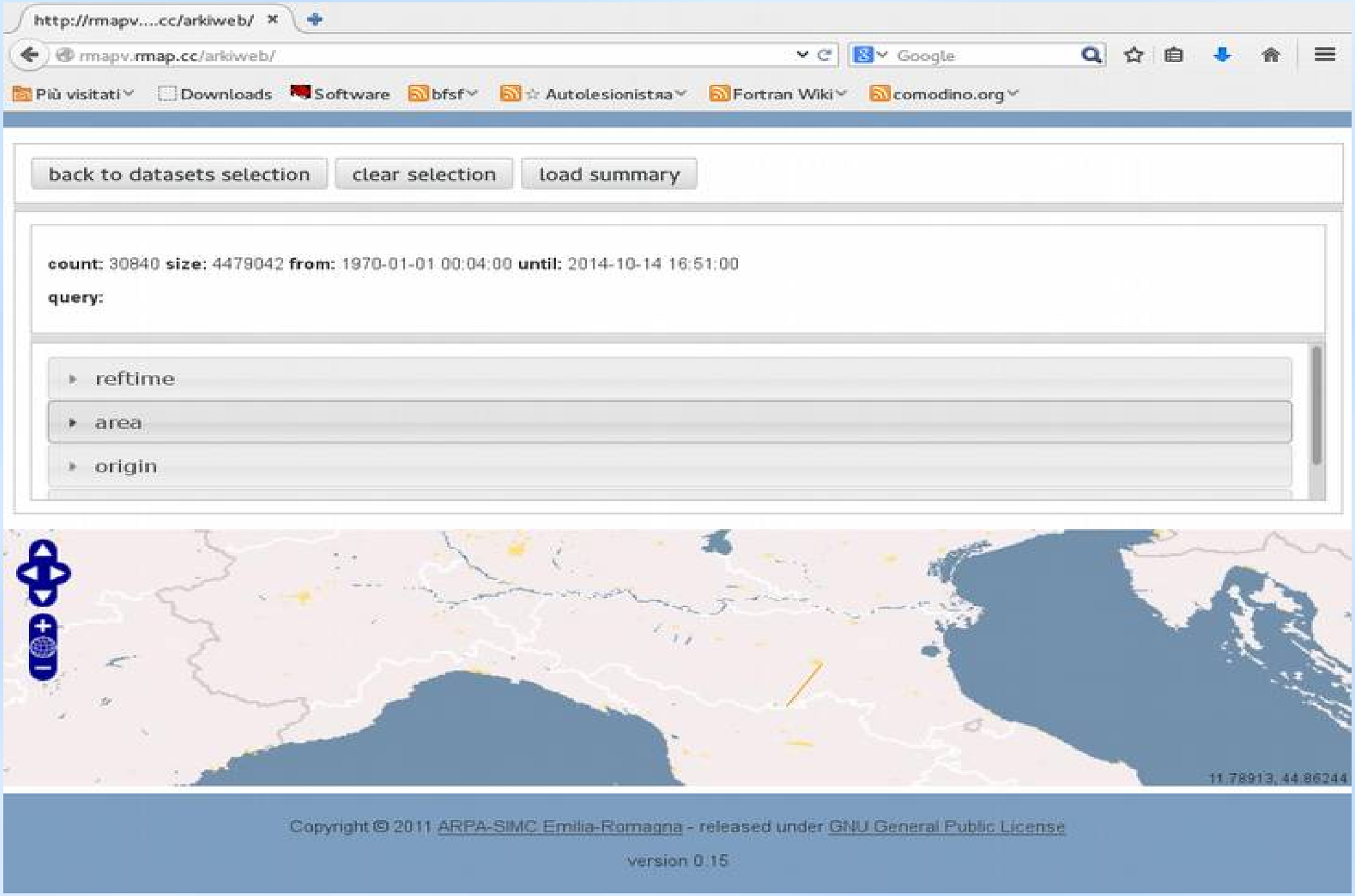
Arkimet: distributed



Borinud: DB-All.e web and web service



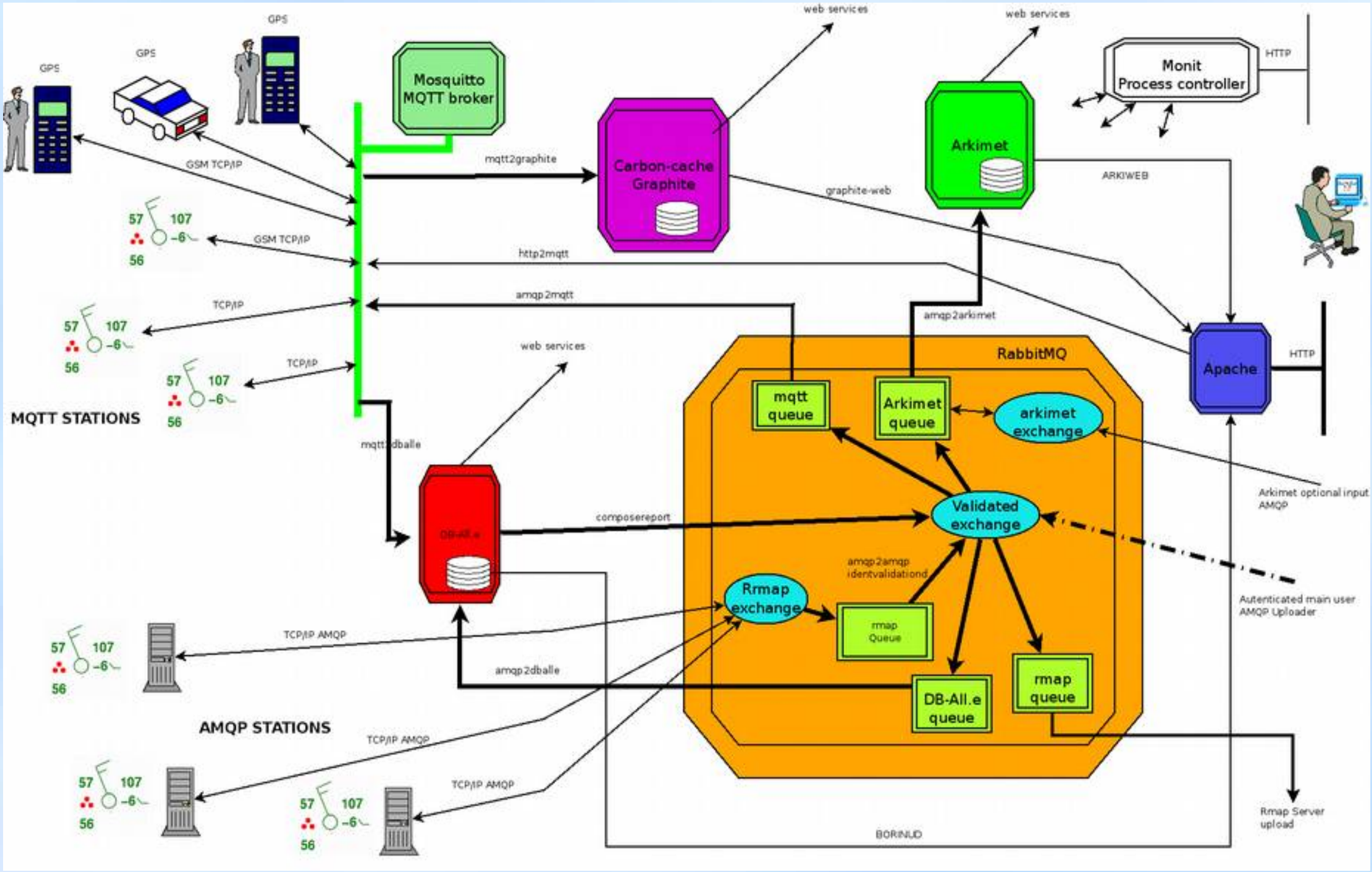
Arkiweb


http://rmapv....cc/arkiweb/
rmapv.rmap.cc/arkiweb/
Google
Più visitati
Downloads
Software
bfsf
Autolesionistяa
Fortran Wiki
comodino.org
back to datasets selection
clear selection
load summary
count: 30840 size: 4479042 from: 1970-01-01 00:04:00 until: 2014-10-14 16:51:00
query:
reftime
area
origin
11.78913, 44.86244
Copyright © 2011 ARPA-SIMC Emilia-Romagna - released under GNU General Public License
version 0.15

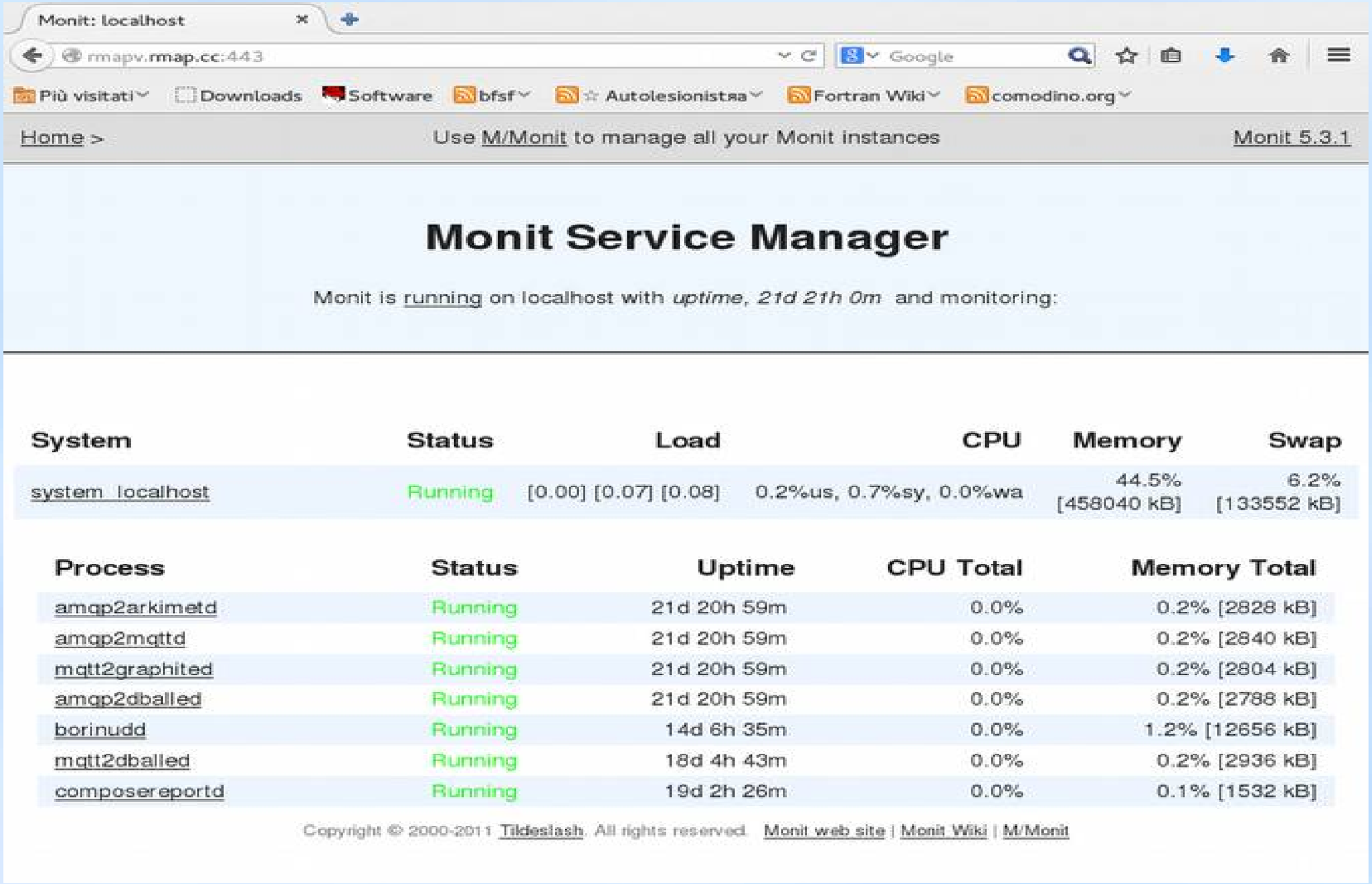
Libsim

- Libsim comprende quattro gruppi di moduli di utilità in Fortran 90:
  - libsim_base definisce moduli e classi di uso generale in applicazioni scientifiche, come la gestione di errori in esecuzione, la gestione di dati georeferenziati, di coordinate temporali, ecc.
  - libsim_grib definisce una serie di classi ad alto livello stratificate sopra la libreria ECMWF grib_api per gestire l'I/O di file in formato grib.
  - libsim_vol7d definisce una serie di classi per facilitare l'elaborazione di dati osservativi idro-meteo, includendo metodi per la loro importazione da database tipo DbAll-e
  - libsim_volgrid6d definisce una serie di classi per facilitare l'elaborazione di dati idro-meteo su grigliati georeferenziati, compresa la trasformazione in griglie di tipo diverso e in oggetti di tipo vol7d.

Flusso dati e processi



Monit


Monit: localhost
rmapv.rmap.cc:443
Google
Più visitati
Downloads
Software
bfsf
Autolesionistaa
Fortran Wiki
comodino.org
Home >
Use M/Monit to manage all your Monit instances
Monit 5.3.1
Monit Service Manager
Monit is running on localhost with uptime, 21d 21h 0m and monitoring:
System
Status
Load
CPU
Memory
Swap
system_localhost
Running
[0.00] [0.07] [0.08]
0.2%us, 0.7%sy, 0.0%wa
44.5% [458040 kB]
6.2% [133552 kB]
Process
Status
Uptime
CPU Total
Memory Total
amqp2arkimetd
Running
21d 20h 59m
0.0%
0.2% [2828 kB]
amqp2mqttd
Running
21d 20h 59m
0.0%
0.2% [2840 kB]
mqtt2graphited
Running
21d 20h 59m
0.0%
0.2% [2804 kB]
amqp2dballed
Running
21d 20h 59m
0.0%
0.2% [2788 kB]
borinudd
Running
14d 6h 35m
0.0%
1.2% [12656 kB]
mqtt2dballed
Running
18d 4h 43m
0.0%
0.2% [2936 kB]
composereportd
Running
19d 2h 26m
0.0%
0.1% [1532 kB]
Copyright © 2000-2011 Tildeslash. All rights reserved. Monit web site | Monit Wiki | M/Monit

Total Physical Source Lines of Code

non-blank, non-comment lines

cost estimates include design, coding, testing (including integration and testing), documentation (both for users and for programmers), and a wrap rate for corporate overhead (to cover facilities, equipment, accounting, and so on)

- Python (stazione + server+GUI)
  - Total Physical Source Lines of Code (SLOC) = **15,626**
  - Total Estimated Cost to Develop = **$ 484,375**
- Firmware Stima
  - Total Physical Source Lines of Code (SLOC) = **131,352**
    - Solo il codice sviluppato per Stima
      - Total Physical Source Lines of Code (SLOC) = **12,763**
      - Total Estimated Cost to Develop = **$ 391,644**
- Mqtt2bufr
  - Total Physical Source Lines of Code (SLOC) = **730**
  - Total Estimated Cost to Develop = $ **19,415**

Il firmware STIMA

Stazioni fisse o mobili

È possibile installare sia stazioni fisse, la cui posizione non cambia nel tempo, sia stazioni mobili, sia terrestri che marine. Per aggiornare la posizioni delle stazioni mobili viene utilizzato un GPS che può essere

- a bordo del modulo Stima
- a bordo di un dispositivo android.

Il firmware STIMA

Differenti tipologie di rete

La configurazione della rete può essere differente a seconda delle esigenze:

- configurazione a stella (moduli master e base) con un broker al centro
- configurazione ad albero sia via cavo (modulo master + base) che via radio

con la possibilità di utilizzare moduli radio di maggiore potenza (~1Km in aria libera) è possibile prevedere coperture di un territorio di ampia superficie.

Il firmware STIMA

Salvataggio locale dei dati

I dati possono essere pubblicati in real time e/o salvati localmente.

- salvataggio dei dati su SD formattata FAT;
- i file vengono frammentati a una dimensione prefissata per farne circa uno al giorno e numerati da 000 a 999; i dati salvati hanno un flag che indica se i dati sono stati già pubblicati correttamente su MQTT;
- i file che devono essere controllati per possibili reinvii hanno postfisso .que e quelli che hanno tutti i dati già inviati hanno postfisso .don.

Il firmware STIMA

Salvataggio locale dei dati; funzionalità:

- salvataggio dati su SD almeno per due anni con campionamenti ogni 5s (un parametro)
- reinvio automatico al server dei dati salvati ma non pubblicati correttamente sul server
- ottimizzazione dei tempi in quanto solo i file che contengono dati da inviare vengono letti per selezionare i dati da reinviare
- i dati possono essere riletti su un normale PC estraendo la SD

Il firmware STIMA

Messaggistica di diagnostica

C'è la possibilità di ottenere una ampia messaggistica di diagnostica per la soluzione dei problemi

attiva di default può essere disabilitata

Il firmware STIMA

Configurazione

Le versioni delle configurazioni vengono verificate e quando il firmware non è retrocompatibile il modulo resta in attesa di una nuova configurazione

Si può forzare la configurazione tramite un apposito ponticello sulla board Stima-I2C

Le configurazioni vengono subito verificate: non è possibile configurare un modulo con dei sensori non corretti o non funzionanti

Il firmware STIMA

Operazioni di mantenimento periodiche

Il software effettua periodicamente tutte le funzioni di mantenimento necessarie a un corretto funzionamento:

- DHCP
- sincronizzazione dell'orologio interno con una sorgente esterna
- la gestione dei pacchetti per il mantenimento dei protocolli su TCP/IP o via radio

Tutti i firmware hanno attivo un watchdog hardware che evita blocchi permanenti dovuti a malfunzionamenti su eventi improbabili; ogni 8 secondi quindi il watchdog deve essere reinizializzato per evitare un reset del microcontrollore.

Il firmware STIMA

Orologio di riferimento

- Una base dei tempi precisa è richiesta nel caso in cui sia necessario salvare i dati localmente (su SD) nel caso la connessione utilizzata per pubblicare i dati sul server (broker) non sia considerata stabile.
- Se invece la connessione (trasporto) viene considerata stabile (o non sia necessario recuperare i dati in caso di fault) un preciso orologio di riferimento non è necessario e il tempo di riferimento verrà aggiunto automaticamente dal server in tempo reale alla pubblicazione del dato.

Ci sono diversi sistemi per avere un orologio di riferimento preciso sui moduli Stima.

Il firmware STIMA

Crittografia

Qualora il trasporto non sia considerato sicuro (via radio) viene utilizzata la crittografia per garantire riservatezza e autenticità.

Per ora il sistema è molto semplice ed utilizza AES con chiavi statiche.

Il firmware STIMA

Attenzione ai consumi energetici

Attenzione è stata posta alla limitazione dei consumi

- Quando possibile i microcontrollori e i sensori vengono messi in sleep e sono alcuni interrupt a risvegliare il sistema

Questo agevola l'utilizzo con batterie dei sistemi a basso consumo quali il modulo Stima-satellite che funziona con un modulo radio.

Il firmware STIMA

Integrazione con la domotica

Per quello che è stato possibile si è cercato di integrarsi con gli standard della domotica (MQTT)

Tutti i moduli possono essere utilizzati anche da attuatori on/off (fino a 4 relay)

è molto semplice aggiungere altre funzionalità tramite remote procedure in formato json su tutti i trasporti

Le librerie utilizzate dal firmware

aJson

aJson è un tentativo di portare una completa implementazione di JSON a Arduino.

E' basata su cJSON, ridotta di dimensione.

La libreria è stata adattata per ridurre l'uso della memoria e implementare alcuni tipi dato non supportati.

Le librerie utilizzate dal firmware

JsonRPC

implementa un sottoinsieme del protocollo JSON-RPC.

PubSubClient

Fornisce un client per semplici publish/subscribe scambiando messaggi con un server che supporta MQTT (broker).

Modificata per funzionare oltre che con ethernet anche con GSM/GPRS.

Le librerie utilizzate dal firmware

Arduino_uip

Implementa le stesse API della Arduino Ethernet library ma utilizzando ENC28J60 come chip per la comunicazione ethernet.

Supporto completo per le connessioni TCP e UDP persistenti (streaming) (Client e Server), ARP, ICMP, DHCP and DNS.

Sviluppata da Norbert Truchsess derivando dallo stack uIP di Adam Dunkels. Oltre a creare una implementazione completamente open permette di utilizzare ENC28J60, chip molto più economico di quelli con stack IP incluso.

Le librerie utilizzate dal firmware

RF24 / RF24Network

ultima versione elaborata da TMRh20 che ci è risultata essere la più stabile e con più opzioni oltre a funzionare con Arduino e Rpi.

- OSI Network Layer ottimizzato per radio nRF24L01(+) 2.4GHz ISM. Con la tipologia di modulo da noi utlizzato si possono coprire distanze di circa 50m in aria libera.
- Host Addressing: ogni nodo ha un indirizzo logico nella rete locale.
- Message Forwarding: i messagi possono essere mandati da un nodo a qualsiasi altro nodo senza limite al numero di "salti" che il messaggio deve fare.
- Ad-hoc Joining: un nodo può entrare a far parte della rete senza nessun cambiamento alla configrazione dei nodi già esistenti.

Le librerie utilizzate dal firmware

sim800

Questa libreria è stata sviluppata ex novo in quanto le funzionalità richieste non sono disponibili in nessun altra libreria di gestione dei moduli sim800/sim900. La libreria implementa:

- TCP/IP transparent mode con le API Etherlib
- http in modalità nativa sim800
- utilizzo dell'RTC interno alla sim800

Le librerie utilizzate dal firmware

SensorDriver

Libreria di "driver" per la gestione dei sensori. Di questa libreria esistono attualmente due versioni, una in C++ e una in python.

Porta la gestione della sensoristica ad un livello di astrazione più alto.

Aggiungere un nuovo tipo di sensore consiste nell'estendere una classe con quattro metodi per effettuare la lettura di quello specifico sensore

int setup (int address);

effettua eventuali settaggi necessari al funzionamento del sensore

int prepare (unsigned long* waittime);

impartisce al sensore il comando per effettuare una singola misurazione torna il tempo in millisecondi di attesa necessario

int get (int* value);

torna i valori della misurazione

aJsonObject* getJson();

torna i valori misurati in formato json

Le librerie utilizzate dal firmware

Time

Time fornisce un orologio software; l'orologio può essere sincronizzato con sorgenti esterne per mantenere l'orologio preciso. Nel nostro caso utilizziamo

- NTP se disponibile ethernet
- RTC della sim800
- RTC del DS1307 tramite I2C.

TimeAlarms

TimeAlarms unitamente a Time esegue funzioni a istanti di tempo specificati, unatantum o periodicamente.

Le librerie utilizzate dal firmware

SdFat

SdFat è una libreria per Arduino che supporta FAT16 and FAT32 file systems su SD cards standard o ad alta capacità (SD/SDHC flash cards)

SdFat supporta la creazione di file, cancellazione, read, write, oltre al troncamento. SdFat supporta l'accesso a subdirectories, creazione, cancellazione si subdirectories. Supporta Long File Names e usa la libreria Arduino SPI

Le librerie utilizzate dal firmware

AESLib

Questa libreria implementa la crittografia tramite AES per arduino.

La crittografia nel nostro firmware è utilizzata per le comunicazioni con nrf24, ma il livello di sicurezza non è ancora elevato e l'implementazione è incompleta.

Le librerie utilizzate dal firmware

YwrobotLiquidCrystal_I2C

Libreria per la gestione del display LCD tramite I2C

Supporta gran parte delle funzioni listate nella specifica ufficiale per i display LCD

Configurazione a tempo di compilazione

- Il firmware Stima fa un uso intensivo delle direttive del **preprocessore C**

- Definendo delle variabili è possibile ottenere da un unico codice diversi firmware per i vari moduli

- Il file **rmap_config.h** situato in sketchbook/rmap/rmap comanda quasi tutte le opzioni e le configurazioni hardware. Sono presenti i **template** da utilizzare per ottenere i firmware per i moduli qui presentati.

Le funzioni principali del firmware

Nel main loop:

- **mgrserialjsonrpc**: gestione della porta seriale per jsonrpc
- **mgrrf24jsonrpc**: gestione della comunicazione radio e jsonrpc
- **mgrmqtt**: gestione del protocollo MQTT e jsonrpc
- **mgrethserver**: gestione del server tcp/ip per jsonrpc

La funzione **mgrjsonrpc** è la funzione comune per gestire il protocollo json-RPC.

Se il modulo è “attivo” periodicamente viene eseguita la funzione **Repeats** che svolge le operazioni di interrogazione dei sensori e pubblicazione dei dati.

STIMA software

- http://rmap.cc
- http://liste.raspibo.org/wws/subscribe/meteo
- https://github.com/r-map