

Sommario

Sommario	1
Specifiche e howto per Stima versione 3	3
• Caratteristiche hardware dei singoli moduli	3
• Stima core+644 e Stima core+1284	3
• Stima I2C-Base	3
• Stima I2C-Digital	5
• Stima I2C-RTC	5
• Stima I2C-FT232RL	6
• Stima SD-Card	7
• Stima SIM800C Power	7
• Stima SIM800C Module	8
• Stima I2C-HUB	9
• Regolatore di tensione e carica della batteria con bus I ² C DigitecoPower	10
• Display LCD 20x4 con interfaccia I2C	12
• Configurazioni hardware dei moduli Stima	13
• Modulo Stima Ethernet	13
• Modulo Stima GSM/GPRS	14
• Modulo Stima Passivo	14
• Modulo Stima I2C-TH	15
• Modulo Stima I2C-Rain	16
• Caratteristiche software	18
• Macchina a stati finiti	18
• Implementazione software della macchina a stati finiti	18
• SensorDriver	20
• Implementazione del corretto funzionamento di dispositivi multi-master su bus I ² C	21
• Microcontrollori e hardware in modalità risparmio energetico	22
• Salvataggio dei dati su SD-Card	22
• Configurazioni software dei moduli Stima	24
• Configurazione dell'IDE Arduino	24
• Modulo Stima Ethernet	24
• Modulo Stima GSM/GPRS	25
• Modulo Stima Passive	25
• Modulo Stima I2C-TH	26
• Modulo Stima I2C-Rain	27

• Compilazione e caricamento del bootloader Digitecoboot	28
• Compilazione del firmware in formato binario	29
• Upload del firmware tramite micro SD-Card attraverso Digitecoboot	30
• Assemblaggio stazione Stima	31

Specifiche e howto per Stima versione 3

- Caratteristiche hardware dei singoli moduli
- Stima core+644 e Stima core+1284



Le due board Stima core+644 e Stima core+1284 sono realizzate sul medesimo circuito stampato, appositamente progettato per accogliere indifferentemente il microcontrollore ATmega644 o ATmega1284 nelle configurazioni a 8 MHz a 3.3V o 16 MHz a 5V.

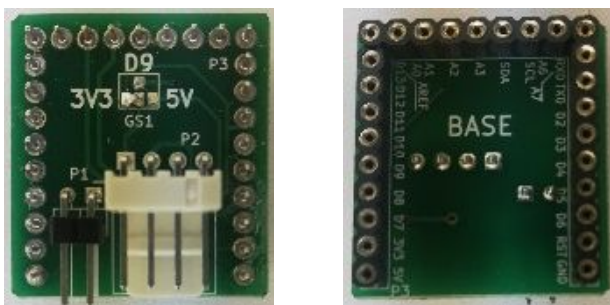
Per selezionare la corretta tensione di alimentazione sul microcontrollore, è sufficiente un punto di saldatura sullo switch delle piste del circuito stampato della scheda, a patto di aver selezionato il giusto quarzo in funzione della tensione di alimentazione.

Entrambe le board sono realizzate sullo standard UPIN-27.

Gli schemi elettrici realizzati in KiCad sono disponibili sul repository github rmap al seguente link:

[kicad/r-map_v3/Microduino_Core+_DE](https://github.com/kicad/r-map_v3/Microduino_Core+_DE)

- Stima I2C-Base



Basato sullo standard UPIN-27 e quindi impilabile come un normalissimo modulo Microduino, è stato progettato per far fronte alle necessità di alimentazione e collegamento su bus I²C dell'intero stack.

È dotato di un connettore a quattro pin, posto nella parte frontale del modulo (l'unico libero dal connettore UPIN-27) la cui funzione duale è quella di alimentare lo stack e di fornire il collegamento su bus I²C a tutti i moduli ad esso collegati.

Per la corretta configurazione hardware di tale modulo, è necessario effettuare un punto di saldatura sullo switch delle piste del circuito stampato atto a selezionare l'alimentazione (3.3V o 5V) e la corrispondente tensione di alimentazione del bus I²C sullo stack di moduli nel quale verrà installato.

Inoltre, tale switch presenta un'ulteriore via che prevede l'abilitazione dell'alimentazione sul bus I²C controllabile attraverso un pin di un microcontrollore impilato nello stack: funzione utile nel caso di test o di configurazione di particolari sensori I²C.

Si specifica che il connettore a quattro pin presenta clip di collegamento elettriche in materiale anti ossidante in grado di garantire un perfetto contatto elettrico nel corso degli anni. Inoltre è realizzato in modo tale da non permettere il distacco accidentale del cavo, assicurando la comunicazione. Altra peculiarità è quella di poter essere innestato in un unico verso evitando l'inversione di polarità di alimentazione e di collegamento del bus.

Tale board, inoltre, presenta due pin cortocircuitabili temporaneamente attraverso un jumper atto a consentire la configurazione della stazione attraverso porta seriale.

A causa dell'ingombro in altezza maggiorato rispetto ad una board standard dovuto al connettore a quattro pin, tale scheda è impilabile solamente attraverso l'adozione di due file di pin del tipo UPIN-27.

Gli schemi elettrici realizzati in KiCad sono disponibili sul repository github rmap al seguente link:

[kicad/r-map_v3/Microduino_BASE_DE](https://github.com/rmap/kicad/r-map_v3/Microduino_BASE_DE)

- Stima I2C-Digital



Il modulo in oggetto, realizza le funzioni di input o output di segnali digitali. Basato sullo standard UPIN-27, sarà dotato di un connettore a quattro pin del tipo usato nel modulo Stima I2C-Base ma adoperato per l'input o l'output di segnali digitali.

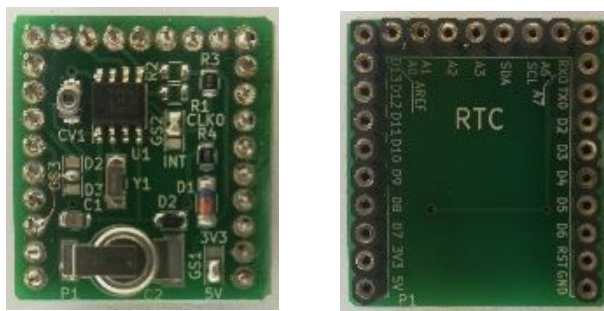
In particolare, sono presenti tre pin corrispondenti alle tre linee di interrupt fisiche fornite dai microcontrollori ed un pin di massa, comune alle tre linee.

A causa dell'ingombro in altezza maggiorato rispetto ad una board standard dovuto al connettore a quattro pin, tale scheda è impilabile solamente attraverso l'adozione di due file di pin del tipo UPIN-27.

Gli schemi elettrici realizzati in KiCad sono disponibili sul repository github rmap al seguente link:

[kicad/r-map_v3/Microduino_DIGITALE_DE](https://github.com/rmap/kicad/r-map_v3/Microduino_DIGITALE_DE)

- Stima I2C-RTC



Tale modulo, compatibile con lo standard UPIN-27, assolve le funzioni di mantenimento della data ed ora nel tempo, attraverso l'adozione del Real Time Clock (RTC), interfacciato al microcontrollore attraverso bus I²C.

Per la corretta configurazione hardware di tale modulo, è necessario effettuare un punto di saldatura sullo switch delle piste del circuito stampato atto a selezionare l'alimentazione (3.3V o 5V).

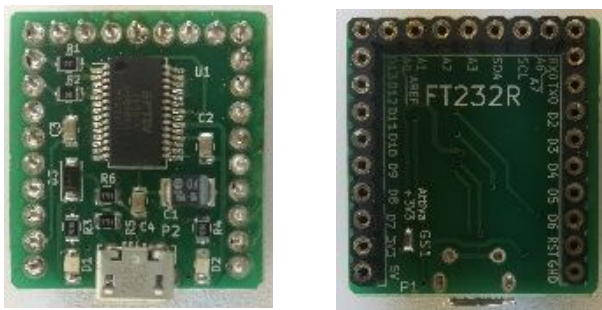
Inoltre, per garantire il funzionamento dello stesso nel caso di distacco dell'alimentazione principale, è provvisto di un super condensatore in grado di fornire energia sufficiente a garantirne il funzionamento per alcune ore.

Un'ulteriore caratteristica del modulo, è quella di poter generare un segnale a frequenza costante e programmabile via software, che potrà essere convogliata su una delle tre linee digitali ad interrupt del microcontrollore, utilizzabile come segnale di risveglio per il microcontrollore impostato in risparmio energetico profondo, in modo tale da poter ridurre i consumi energetici dell'intera stazione.

Gli schemi elettrici realizzati in KiCad sono disponibili sul repository github rmap al seguente link:

[kicad/r-map_v3/Microduino_RTC_DE](https://github.com/rmap/kicad/tree/master/r-map_v3/Microduino_RTC_DE)

- **Stima I2C-FT232RL**



Il modulo in questione, permette di collegare la porta seriale zero del microcontrollore ad un bus USB (ad esempio, quello di un computer) in modo tale da poter facilmente instaurare una comunicazione seriale con il microcontrollore. Tale metodo di comunicazione risulta comodo sia nel caso di upload del firmware che nel caso di configurazione della stazione.

Inoltre, l'integrato FT232RL è in grado di generare una tensione a 3.3V a bassa potenza in grado di alimentare direttamente il microcontrollore nella configurazione 3.3V e 8MHz, permettendone la programmazione dello stesso quando non è alimentato da una fonte di alimentazione esterna.

Tale tensione può o meno essere fornita al microcontrollore effettuando o meno una saldatura sul circuito stampato della board. L'intero modulo è realizzato secondo lo standard UPIN-27.

Gli schemi elettrici realizzati in KiCad sono disponibili sul repository github rmap al seguente link:

[kicad/r-map_v3/Microduino_FT232R_DE](https://github.com/rmap/kicad/r-map_v3/Microduino_FT232R_DE)

- Stima SD-Card

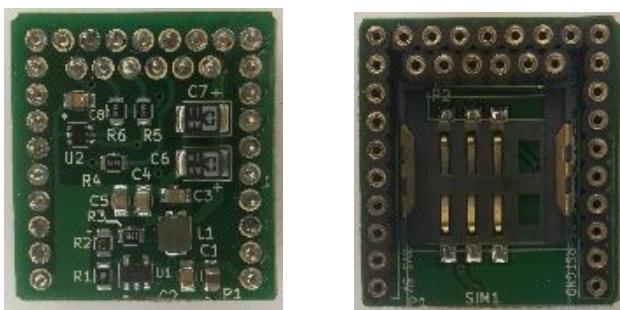


Tale board permette il collegamento di una micro SD-Card al microcontrollore permettendo il salvataggio dei dati dei sensori e l'upload del firmware con il bootloader Digitecboot. L'intera board è anch'essa basata sullo standard UPIN-27.

Gli schemi elettrici realizzati in KiCad sono disponibili sul repository github rmap al seguente link:

[kicad/r-map_v3/Microduino_SD_DE](https://github.com/rmap/kicad/r-map_v3/Microduino_SD_DE)

- Stima SIM800C Power



Il modulo SIM800C Power costituisce il modulo di alimentazione per il modulo GSM/GPRS SIM800C ed include il connettore per la SIM Card in formato standard. Basato sul connettore UPIN-27, necessita di ulteriori 6 pin posti a contatto con la parte posteriore del connettore principale, atti al trasporto di tensioni di alimentazione e segnali verso il modulo SIM800C posto nella board Stima SIM800C Module.

Il comparto di alimentazione è realizzato mediante alimentatore switching e fornisce la giusta alimentazione al modulo SIM800C a partire dalla tensione di alimentazione primaria di 5V.

A causa dell'ingombro in altezza maggiorato rispetto ad una board standard dovuto alla componentistica elettronica, tale scheda è impilabile solamente attraverso l'adozione di una fila di pin del tipo UPIN-27.

Gli schemi elettrici realizzati in KiCad sono disponibili sul repository github rmap al seguente link:

[kicad/r-map_v3/Microduino_GSM_DE/SIM_shield_DE](https://github.com/rmap/kicad/r-map_v3/Microduino_GSM_DE/SIM_shield_DE)

- **Stima SIM800C Module**



In tale modulo basato sul connettore UPIN-27, trova alloggio il modulo GSM/GPRS SIM800C, due led di funzionamento ed il connettore SMA per il collegamento dell'antenna, introdotto in quanto molto più solido, robusto e di facile installazione rispetto a quello adottato sulle board Microduino.

Per garantire il risparmio energetico della stazione, il modulo GSM/GPRS è sempre spento e viene acceso solo nel momento in cui si rende necessario il suo utilizzo. L'operazione di accensione o spegnimento è effettuata ponendo a livello logico basso il pin dedicato del modulo per almeno un secondo. L'identificazione di tale pin è inserita nel file `gsm_config.h` nella cartella `arduino/sketchbook/libraries/RmapConfig`.

Il modulo è provvisto di due led di stato che forniscono indicazioni sul funzionamento e sullo stato della rete. In particolare, osservando il modulo frontalmente, il led di sinistra:

- Acceso: il modulo è acceso ed è in funzionamento
- Spento: il modulo è spento

Il led di destra:

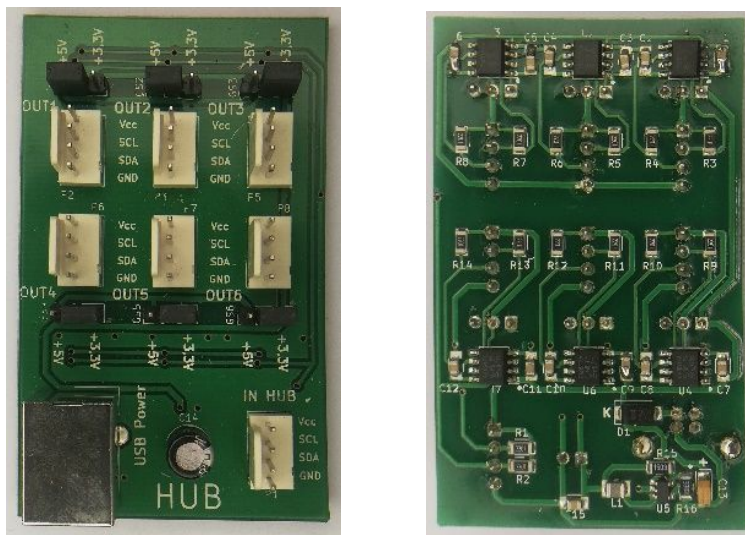
- Spento: il modulo è spento
- Acceso per 64ms e spento per 800ms: il modulo non è registrato sulla rete
- Acceso per 64ms e spento per 3000ms: il modulo è registrato sulla rete
- Acceso per 64ms e spento per 300ms: comunicazione GPRS in corso

Si consiglia quindi di impilare per prima la board Stima SIM800C Power, di interporre un'ulteriore fila di pin del connettore UPIN-27 e di impilare in testa la board Stima SIM800C Module.

Gli schemi elettrici realizzati in KiCad sono disponibili sul repository github rmap al seguente link:

[kicad/r-map_v3/Microduino_GSM_DE/SIM800C_DE](https://github.com/rmap/kicad/r-map_v3/Microduino_GSM_DE/SIM800C_DE)

- **Stima I2C-HUB**



Il modulo Stima I2C-HUB garantisce la massima modularità dell'intero progetto della stazione meteorologica Stima, preservando la compatibilità con tutti i moduli Stima esistenti e garantendo il collegamento su bus I²C di dispositivi operanti a tensioni di 3.3 V e 5 V contemporaneamente.

L'HUB presenta una porta in ingresso e sei porte in uscita, utilizzabili per il collegamento e/o alimentazione di moduli su bus I²C. Ogni porta è realizzata mediante il connettore a quattro pin dello stesso tipo di quello installato negli altri moduli (Stima I2C-Base e Stima I2C-Digital).

Fatta eccezione per la porta in ingresso, la quale dev'essere sempre collegata ad un bus I²C a 5V, ognuna delle sei porte in uscita è configurabile indipendentemente per collegare e/o alimentare moduli funzionanti a tensioni di 3.3V o 5V, selezionabili facilmente e velocemente attraverso un jumper posto in prossimità di ogni porta. Inoltre, ogni singola porta è bufferizzata e protetta contro scariche elettrostatiche.

L'HUB è in grado di ricevere alimentazione a 5V attraverso il connettore USB tipo B o dalla porta in ingresso con connettore a quattro pin. Tale alimentazione, verrà propagata sull'intera linea a 5V e su tutti i moduli connessi all'HUB.

Diversamente, un circuito di alimentazione da 5V a 3.3V posto sull'HUB, provvederà ad alimentare tutti i moduli collegati sul bus a 3.3V.

Con riferimento alla porta in ingresso con connettore a quattro pin, si evidenzia che tale porta può essere adottata per collegare un HUB B ad un ad HUB A nel caso in cui vi sia la necessità di collegare un certo numero di periferiche sull'HUB B posto a distanza dall'HUB A. In tal caso, sarà sufficiente collegare attraverso un cavo, una delle sei porte in uscita dell'HUB A impostata a 5V al connettore in ingresso dell'HUB B.

Stima I2C-HUB consente la piena modularità della stazione Stima, garantendo molteplici topologie di collegamento dei moduli o dei sensori, facilitandone la connessione ed il cablaggio e dimenticandosi di calcolare di volta in volta i giusti valori di resistenze di pull-up e il massimo limite di capacità del bus I²C.

Gli schemi elettrici realizzati in KiCad sono disponibili sul repository github rmap al seguente link:

[kicad/r-map_v3/Microduino_HUB_DE](https://github.com/rmap/kicad/r-map_v3/Microduino_HUB_DE)

- **Regolatore di tensione e carica della batteria con bus I²C DigitecoPower**

Il regolatore di alimentazione DigitecoPower, di tipo switching, è progettato per garantire un uso efficiente della batteria con funzionamento in tampone.

Nel caso specifico, il regolatore monitorerà lo stato di salute della batteria fornendo i dati riassuntivi tramite interrogazione su bus I²C. È provvisto di una linea di alimentazione 12-30 V DC fornita da un pannello fotovoltaico o da un alimentatore DC, di una linea a 12V in uscita per la connessione di una batteria al piombo di tipo sigillato e di quattro pin per la connessione di tale alimentatore alla porta a quattro pin in ingresso al HUB.

In particolare, i valori monitorabili ed acquisiti attraverso la libreria DigitecoPower implementata nella libreria SensorDriver, sono:

- Tensione di alimentazione
- Corrente di alimentazione
- Tensione di batteria
- Corrente di batteria
- Percentuale di carica della batteria
- Tensione a 5V

Per la verifica istantanea del funzionamento, è presente un led di notifica tri-colore (rosso, giallo, verde) in grado di indicare la carica o meno della batteria ed il livello di energia della stessa:

- Led rosso: batteria scarica
- Led giallo: batteria mediamente carica
- Led verde: batteria carica
- Led lampeggiante lentamente: batteria in scarica
- Led lampeggiante velocemente: batteria in carica

Ad esempio, un led rosso con un lampeggio veloce indica che la batteria è scarica e si sta caricando, diversamente, un led verde con lampeggio lento, indica che la batteria è carica ma si sta scaricando.

Per l'interfacciamento di tale modulo, si faccia riferimento al seguente schema di collegamento:

- 1) VCC_IN: ingresso alimentazione 12-30V DC VCC (+)
- 2) GND_IN: ingresso alimentazione 12-30V DC GND (-)
- 3) VCC_BAT: uscita alimentazione batteria 12V DC VCC (+)
- 4) GND_BAT: uscita alimentazione batteria 12V DC GND (-)
- 5) LED di stato
- 6) VCC_OUT: uscita tensione di alimentazione HUB 5V DC VCC (+)
- 7) SCL: I2C SCL HUB
- 8) SDA: I2C SDA HUB
- 9) GND_OUT: uscita tensione di alimentazione HUB 5V DC GND (-)

- Display LCD 20x4 con interfaccia I2C

Il display consente di visualizzare lo stato della stazione: prossimo orario di acquisizione, stato del salvataggio dei dati su sd-card, stato dell'invio dei dati attraverso mqtt ed un valore rappresentativo dell'ultima acquisizione, utile per verificare in campo il corretto funzionamento della strumentazione.

Inoltre, è provvisto di un pulsante per l'accensione temporanea della retroilluminazione, evitando inutilmente il consumo della batteria.

- Configurazioni hardware dei moduli Stima

- Modulo Stima Ethernet

Il modulo Stima Ethernet consente la connessione della stazione attraverso Ethernet con eventuale alimentazione PoE ed è realizzabile assemblando i seguenti moduli (dal basso verso l'alto):

1. Stima I2C-Base @ 5V
2. Microduino Ethernet WIZ
3. Microduino RJ45
4. Stima core+1284 @ 5V
5. Stima I2C-RTC @ 5V
6. Stima FT232RL
7. Stima SD-Card



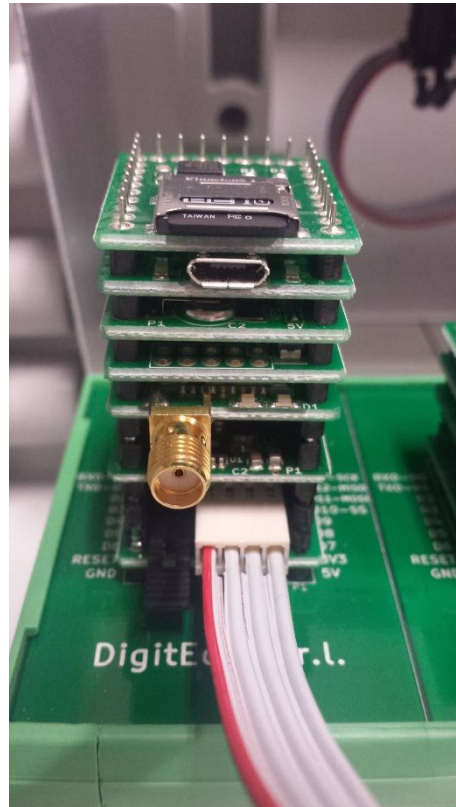
Successivamente sarà necessario fornire connettività Ethernet e alimentazione attraverso una delle seguenti modalità:

- Power over ethernet (PoE)
- Connettore micro USB posto sulla board Stima FT232RL
- Connettore a quattro pin posto sulla board Stima I2C-Base
- Collegando il modulo a Stima I2C-HUB

- **Modulo Stima GSM/GPRS**

Il modulo Stima GSM/GPRS consente la connessione della stazione attraverso GPRS ed è realizzabile assemblando i seguenti moduli (dal basso verso l'alto):

1. Stima I2C-Base @ 5V
2. Stima SIM800C Power
3. Stima SIM800C Module
4. Stima core+1284 @ 5V
5. Stima I2C-RTC @ 5V
6. Stima FT232RL
7. Stima SD-Card



Infine, è necessario collegare un'antenna GPRS Dual Band 850/1900 MHz attraverso il connettore SMA posto sulla board Stima SIM800C Module e fornire alimentazione attraverso una delle seguenti modalità:

- Connettore micro USB posto sulla board Stima FT232RL
- Connettore a quattro pin posto sulla board Stima I2C-Base
- Collegando il modulo a Stima I2C-HUB

- **Modulo Stima Passivo**

Tale modulo passivo permette di acquisire i sensori collegati su bus I²C e di esporre i dati ad un altro modulo "attivo" attraverso Remote Procedure Call (RPC) ed è realizzabile assemblando i seguenti moduli (dal basso verso l'alto):

1. Stima I2C-Base @ 5V / 3.3V
2. Stima core+1284 @ 5V / Stima core+644 @ 3.3V
3. Stima I2C-RTC @ 5V / 3.3V
4. Stima FT232RL

A tale modulo, è possibile fornire alimentazione attraverso una delle seguenti modalità:

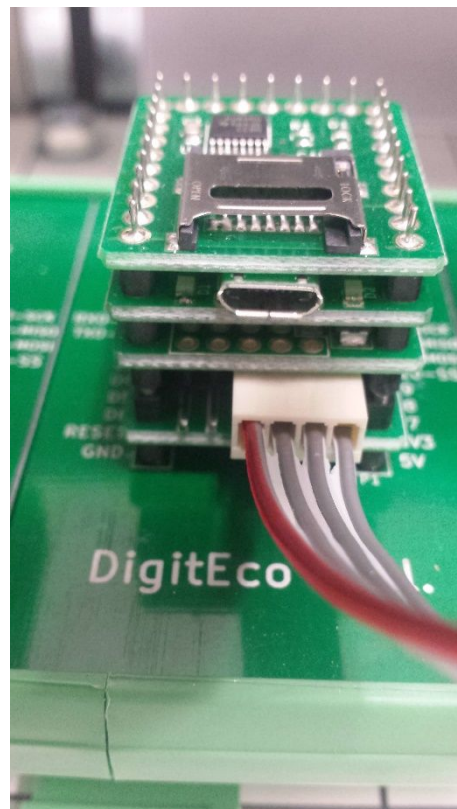
- Connettore micro USB posto sulla board Stima FT232RL
- Connettore a quattro pin posto sulla board Stima I2C-Base
- Collegando il modulo a Stima I2C-HUB

- **Modulo Stima I2C-TH**

Acquisisce i dati dai sensori di temperatura e umidità collegati su bus I²C fornendo elaborazioni relative a valori istantanei, minimi, medi e massimi. Tali dati potranno essere acquisiti da uno dei moduli atti a tale operazione, attraverso la libreria SensorDriver.

Nello specifico, assemblare il modulo con le seguenti board (dal basso verso l'alto):

1. Stima I2C-Base @ 3.3V
2. Stima core+644 @ 3.3V
3. Stima FT232RL
4. Stima SD-Card



Per garantire il risparmio energetico della stazione, il modulo Stima I2C-TH è perennemente posto in risparmio energetico sufficiente al

mantenimento del funzionamento del clock della CPU e del timer ad elevata precisione posto all'interno del microcontrollore.

Il timer è programmato per generare un interrupt allo scadere di un certo intervallo di tempo in secondi (programmabile via software), corrispondente all'intervallo di acquisizione dei sensori di temperatura e umidità.

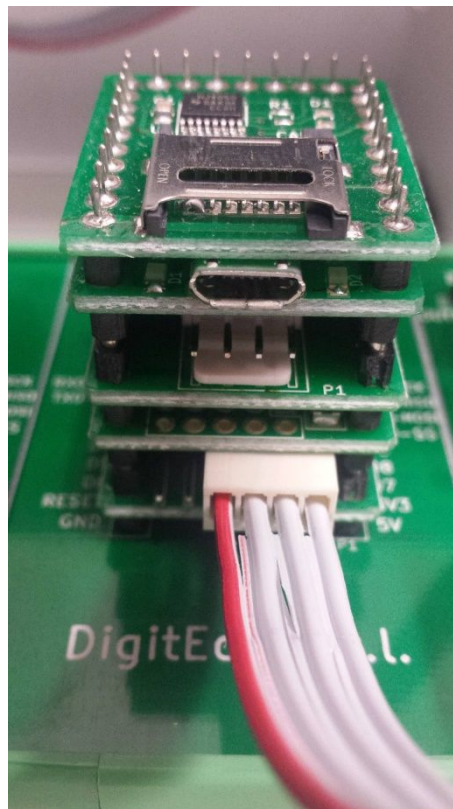
Al termine delle operazioni di acquisizione dei sensori, il microcontrollore torna in risparmio energetico attendendo il successivo istante di acquisizione.

- **Modulo Stima I2C-Rain**

Effettua un conteggio del numero di basculate di un pluviometro nell'arco di tempo desiderato. Tale dato sarà acquisito da uno dei moduli atti a tale operazione, attraverso la libreria SensorDriver.

Nello specifico, assemblare il modulo con le seguenti board (dal basso verso l'alto):

1. Stima I2C-Base @ 3.3V
2. Stima core+644 @ 3.3V
3. Stima I2C-Digital
4. Stima FT232RL
5. Stima SD-Card



In genere la segnalazione di una basculata del pluviometro viene fornita con la chiusura di un contatto elettrico. Tale collegamento può essere realizzato mediante collegamento al pin di massa

e ad una delle tre linee di interrupt presenti sul connettore a quattro pin sulla board Stima I2C-Digital.

Per garantire il risparmio energetico della stazione, il modulo Stima I2C-Rain è perennemente posto in risparmio energetico profondo e risvegliato all'occorrenza da un interrupt scaturito dalla basculata del pluviometro.

- **Caratteristiche software**

- **Macchina a stati finiti**

L'intero software è stato riscritto e sviluppato secondo il modello della macchina a stati finiti ed in particolare, secondo un automa di Mealy in cui la transizione da uno stato ad un altro dipende dallo stato attuale e da eventi esterni. Tale approccio consente di specificare con chiarezza le transizioni da eseguire tra gli stati in base agli eventi ed evitare stati di incoerenza causanti il malfunzionamento o il blocco del sistema.

Ogni sequenza di operazioni è chiaramente identificata e modellata attraverso stati ben precisi. L'implementazione dell'intero automa è realizzata attraverso una variabile indicante lo stato corrente, aggiornata ad ogni passaggio di stato. La scelta dell'esecuzione di un particolare stato avviene per merito di un costrutto switch il quale, ciclo dopo ciclo, processando la variabile indicante lo stato corrente, eseguirà il codice relativo. Tale codice è implementato attraverso l'uso di funzioni non bloccanti: tale approccio consente uno pseudo parallelismo del codice, evitando di assegnare il microcontrollore ad una specifica esecuzione per un periodo di tempo eccessivo tale da penalizzare l'esecuzione di altre funzioni.

La scrittura del software mediante le regole appena descritte consentono un'assoluta modularità ed una rapida scalabilità dello stesso con l'aggiunta di funzionalità che in seguito potrebbero essere richieste.

Inoltre, tale approccio è in piena sintonia con la filosofia del progetto Stima, rendendo l'intero software facilmente comprensibile a chiunque abbia voglia di realizzare la propria stazione meteorologica, in accordo all'idea che sta alla base dell'open source e delle specifiche tecniche RMAP.

- **Implementazione software della macchina a stati finiti**

Ogni task del sistema è composto da:

1. Metodo implementativo delle funzionalità
2. Variabile di stato
3. Variabile booleana indicante se il task è in esecuzione

Per implementare un ipotetico task di esempio, è necessario creare:

1. Una variabile globale booleana indicante se il task è in esecuzione:

```
bool is_event_esempio = false;
```

2. Un nuovo tipo di variabile definendo i vari stati necessari ad implementare le funzionalità del task, come enumerazioni:

```
typedef enum {  
    ESEMPIO_INIT,  
    .  
    ESEMPIO_END,  
    ESEMPIO_WAIT_STATE  
} esempio_state_t;
```

3. Una variabile globale del tipo appena definito:

```
esempio_state_t esempio_state = ESEMPIO_INIT;
```

4. La funzione implementante il task:

```
void esempio_task () {  
    static esempio_state_t state_after_wait;  
    static uint32_t delay_ms;  
    static uint32_t start_time_ms;  
  
    switch (esempio_state) {  
        case ESEMPIO_INIT:  
            state_after_wait = ESEMPIO_INIT;  
            esempio_state = "stato successivo";  
            break;  
  
        .  
        .  
        .  
  
        case ESEMPIO_END:  
            noInterrupts();  
            is_event_esempio = false;  
            ready_tasks_count--;  
            interrupts();  
            esempio_state = ESEMPIO_INIT;  
            break;  
  
        case ESEMPIO_WAIT_STATE:  
            if (millis() - start_time_ms > delay_ms) {  
                esempio_state = state_after_wait;  
            }  
            break;  
    }  
}
```

Se nel corso dell'esecuzione del task è necessario attendere un certo intervallo di tempo attraverso lo stato di attesa non bloccante è possibile farlo mediante:

```
delay_ms = 10;
```

```
start_time_ms = millis();
state_after_wait = "stato successivo allo scadere del timeout di 10 ms";
esempio_state = ESEMPIO_WAIT_STATE;
```

La chiamata al task viene fatta nel loop() e implementata mediante la forma:

```
if (is_event_esempio) {
    esempio_task();
    wdt_reset();
}
```

Per attivare il task in un punto qualsiasi del codice, è possibile adottare la forma:

```
if (!is_event_esempio) {
    noInterrupts();
    is_event_esempio = true;
    ready_tasks_count++;
    interrupts();
}
```

- **SensorDriver**

SensorDriver è la libreria scritta in C++ OOP che implementa la lettura dei sensori attraverso interfacce standard su bus I²C.

Per la lettura dei sensori, viene creato un array del tipo SensorDriver *sensors[COUNT] a cui ad ogni elemento dell'array corrisponde un oggetto di tipo SensorDriver che implementa i metodi descritti nel seguito.

- SensorDriver(const char* driver, const char* type)
 - o Costruttore
 - o const char* driver: stringa di 3 caratteri contenente il nome del driver
 - o const char* type: stringa di 3 caratteri contenente il nome del sensore
- virtual void setup(const uint8_t address, const uint8_t node = 0)
 - o operazioni di inizializzazione del sensore
 - o const uint8_t address: indirizzo I²C del sensore
 - o const uint8_t node: nodo all'interno della rete
- virtual void prepare(bool is_test = false)
 - o inizializzazione del sensore precedente alla lettura
 - o bool is_test: se false il sensore viene preparato per effettuare le normali procedura di lettura, se true il sensore si predispone per leggere valori particolari "di test" utili alla verifica di funzionamento dello stesso

- `virtual void get(int32_t *values, uint8_t length)`
 - o lettura dei valori dal sensore in formato numerico intero a 32 bit con segno
 - o `int32_t *values`: puntatore all'array di ritorno dei valori
 - o `uint8_t length`: numero di valori da leggere dal sensore

- `virtual void getJson(int32_t *values, uint8_t length, char *json_buffer, size_t json_buffer_length = JSON_BUFFER_LENGTH)`
 - o lettura dei valori dal sensore in formato JSON
 - o `int32_t *values`: puntatore all'array di ritorno dei valori
 - o `uint8_t length`: numero di valori da leggere dal sensore
 - o `char *json_buffer`: buffer di ritorno della stringa contenente il JSON
 - o `size_t json_buffer_length`: lunghezza del buffer

- `static SensorDriver *create(const char* driver, const char* type)`
 - o crea un'istanza di `SensorDriver` per un sensore specifico
 - o `const char* driver`: stringa di 3 caratteri contenente il nome del driver
 - o `const char* type`: stringa di 3 caratteri contenente il nome del sensore

- `static void createAndSetup(const char* driver, const char* type, const uint8_t address, const uint8_t node, SensorDriver *sensors[], uint8_t *sensors_count)`
 - o richiama in sequenza i metodi `create` e `setup` assegnando la nuova istanza del sensore all'array delle istanze dei sensori incrementandone la variabile corrispondente che ne indica la dimensione
 - o `const char* driver`: stringa di 3 caratteri contenente il nome del driver
 - o `const char* type`: stringa di 3 caratteri contenente il nome del sensore
 - o `const uint8_t address`: indirizzo I²C del sensore
 - o `uint8_t node`: nodo all'interno della rete
 - o `const u SensorDriver *sensors[]`: array delle istanze dei sensori
 - o `uint8_t *sensors_count`: numero di istanze create

- Implementazione del corretto funzionamento di dispositivi multi-master su bus I²C

L'installazione di molteplici dispositivi in modalità master su un unico bus I²C, comporta l'esplicita gestione della modalità multi-master del bus. Nel particolare caso in oggetto, la gestione multi-master è stata realizzata mediante politica tipo CSMA/CA in cui ogni dispositivo che ha la necessità di acquisire il bus, esegue come prima operazione l'accesso allo stesso in "lettura" e qualora risultasse libero, potrà accedervi in modalità master per l'esecuzione delle operazioni richieste. Qualora il bus risultasse occupato, è necessario attendere un intervallo di tempo della durata casuale ma inferiore ad un dato valore limite, per effettuare un nuovo tentativo di accesso al bus.

Tale metodica è stata implementata modificando la libreria Wire di Arduino.

- Microcontrollori e hardware in modalità risparmio energetico

Per garantire il funzionamento della stazione con batteria e pannello fotovoltaico, i microcontrollori sono impostati in modalità a basso consumo. Tale modalità è raggiunta con lo spegnimento fisico di tutta la strumentazione non strettamente necessaria che sarà alimentata solo nel momento in cui risulti utile (ad esempio: il modulo GSM/GPRS ed alcune periferiche dei microprocessori).

In particolare i moduli Stima Ethernet o Stima GSM/GPRS sono posti in modalità power down e risvegliati con interrupt dell'RTC con cadenza del secondo.

Analogamente, il modulo Stima I2C-Rain è risvegliato dall'interrupt dovuto ad una basculata del pluviometro e il modulo Stima I2C-TH viene svegliato tramite interrupt del timer interno.

Entrambi i moduli Stima I2C-Rain e Stima I2C-TH possono essere risvegliati attraverso matching dell'indirizzo I²C del microcontrollore. Ciò consente di porre tutta la strumentazione in modalità risparmio energetico e qualora un qualsiasi dispositivo multi-master sul bus, si risvegli autonomamente o in seguito ad un evento esterno (esempio: segnalazione di pioggia dal pluviometro), potrà risvegliare tutti i moduli multi-master necessari, con un semplice indirizzamento I²C del dispositivo specifico.

- Salvataggio dei dati su SD-Card

Tutti i dati acquisiti e le relative ed eventuali elaborazioni effettuate, sono salvate su scheda SD-Card e conseguentemente inviati al server RMAP.

Per assolvere tali funzioni ed ottimizzare il funzionamento complessivo della stazione meteorologica in merito ad overhead del tempo di cpu per la ricerca dei file ed all'uso dello spazio sul disco, viene creato un file per ogni giorno di registrazione di dati, salvando all'interno tutti i dati dei sensori relativi a quel giorno.

Per gestire la modalità di invio dati al server, è presente un unico file in cui viene scritto di volta in volta, il puntatore corrispondente alla data ed ora dell'ultimo dato trasmesso al server RMAP. Per effettuare un nuovo trasferimento dei dati a partire da una specifica data antecedente a quella del puntatore ai dati correnti, è sufficiente un aggiornamento di tale puntatore con la data desiderata: sarà compito del software ricercare il primo dato utile successivo a tale data.

Nello specifico, ogni file di dati assume il nome nel formato `aaaa_mm_gg.txt` in cui:

- `aaaa`: anno con 4 cifre
- `mm`: mese con 2 cifre
- `gg`: giorno con 2 cifre


In ogni file, ogni riga corrisponde ad un dato di un sensore ed in particolare, ogni riga è della lunghezza di `MQTT_SENSOR_TOPIC_LENGTH + MQTT_MESSAGE_LENGTH` bytes. Tali valori sono delle `#define` situate nel file `mqtt_config.h` nella cartella `arduino/sketchbook/libraries/RmapConfig`.

Ogni riga è salvata nel formato: `TRANGE/LEVEL/VAR {"v": VALUE, "t": TIME}`

Il file contenente il puntatore all'ultimo dato trasmetto assume il nome `mqtt_ptr.txt` e contiene un dato binario della dimensione di 4 bytes senza segno corrispondente al numero di secondi dal 00:00:00 del 01/01/1970 indicante la data ed ora dell'ultimo dato trasmetto attraverso MQTT.

- Configurazioni software dei moduli Stima

- Configurazione dell'IDE Arduino

Per la compilazione del software è necessario settare lo sketchbook del progetto rmap nell'ide Arduino cliccando su File  Impostazioni e selezionando la cartella dello sketchbook facendo click su sfoglia in alto a destra. Successivamente cliccare su ok, chiudere l'ide per poi riaprirlo.

- Modulo Stima Ethernet

1. Aprire lo sketch rmap.ino nella cartella arduino/sketchbook/rmap/rmap
2. Aprire il file rmap-config.h arduino/sketchbook/rmap/rmap:
 - Impostare la #define MODULE_TYPE con il valore STIMA_MODULE_TYPE_REPORT_ETH per definire una stazione di tipo report o STIMA_MODULE_TYPE_SAMPLE_ETH per definire una stazione di tipo sample
3. Aprire il file sensors_config.h nella cartella arduino/sketchbook/libraries/RmapConfig:
 - Settare la #define USE_JSON con il valore true
 - Settare le altre #define USE_SENSOR_XXX con valori true o false a seconda dei sensori che si vuole abilitare o disabilitare
 - Impostare il valore in decimi di mm di pioggia per ogni basculata del pluviometro nella #define RAIN_FOR_TIP
 - Impostare il numero di valori da leggere per ogni singolo sensore nella #define VALUES_TO_READ_FROM_SENSOR_COUNT (tra tutti i valori a disposizione, inserire quello più alto)
 - Impostare il valore in minuti per il calcolo di un'osservazione nella #define OBSERVATIONS_MINUTES (in genere 1)
 - Impostare il numero di osservazioni necessarie a produrre un dato utile per il report nella #define STATISTICAL_DATA_COUNT (in genere 15)
4. Impostare il microcontrollore su core+1284 a 5V
5. Compilare il firmware e caricarlo attraverso il cavo USB
6. Inserire il jumper sulla board Stima I2C-Base, configurare la stazione e rimuovere il jumper a configurazione terminata.

Per la configurazione è possibile fare riferimento al seguente link:

http://www.raspibo.org/wiki/index.php?title=Gruppo_Meteo/HowTo#Configurazione_5

- **Modulo Stima GSM/GPRS**

1. Aprire lo sketch rmap.ino nella cartella arduino/sketchbook/rmap/rmap
2. Aprire il file rmap-config.h arduino/sketchbook/rmap/rmap:
 - Impostare la #define MODULE_TYPE con il valore STIMA_MODULE_TYPE_REPORT_GSM per definire una stazione di tipo report o STIMA_MODULE_TYPE_SAMPLE_GSM per definire una stazione di tipo sample
3. Aprire il file sensors_config.h nella cartella arduino/sketchbook/libraries/RmapConfig:
 - Settare la #define USE_JSON con il valore true
 - Settare le altre #define USE_SENSOR_XXX con valori true o false a seconda dei sensori che si vuole abilitare o disabilitare
 - Impostare il valore in decimi di mm di pioggia per ogni basculata del pluviometro nella #define RAIN_FOR_TIP
 - Impostare il numero di valori da leggere per ogni singolo sensore nella #define VALUES_TO_READ_FROM_SENSOR_COUNT (tra tutti i valori a disposizione, inserire quello più alto)
 - Impostare il valore in minuti per il calcolo di un'osservazione nella #define OBSERVATIONS_MINUTES (in genere 1)
 - Impostare il numero di osservazioni necessarie a produrre un dato utile per il report nella #define STATISTICAL_DATA_COUNT (in genere 15)
4. Aprire il file gsm_config.h nella cartella arduino/sketchbook/libraries/RmapConfig:
 - Settare la #define GSM_DEFAULT_APN con il valore dell'APN o con il valore di una delle #define predefinite ed incluse nel medesimo file
5. Impostare il microcontrollore su core+1284 a 5V
6. Compilare il firmware e caricarlo attraverso il cavo USB
7. Inserire il jumper sulla board Stima I2C-Base, configurare la stazione e rimuovere il jumper a configurazione terminata

Per la configurazione è possibile fare riferimento al seguente link:

http://www.raspibo.org/wiki/index.php?title=Gruppo_Meteo/HowTo#Configurazione_4

- **Modulo Stima Passive**

1. Aprire lo sketch rmap.ino nella cartella arduino/sketchbook/rmap/rmap

2. Aprire il file rmap-config.h arduino/sketchbook/rmap/rmap:
 - Impostare la #define MODULE_TYPE con il valore STIMA_MODULE_TYPE_PASSIVE per definire una stazione di tipo passivo
3. Aprire il file sensors_config.h nella cartella arduino/sketchbook/libraries/RmapConfig:
 - Settare la #define USE_JSON con il valore true
 - Settare le altre #define USE_SENSOR_XXX con valori true o false a seconda dei sensori che si vuole abilitare o disabilitare
 - Impostare il valore in decimi di mm di pioggia per ogni basculata del pluviometro nella #define RAIN_FOR_TIP
 - Impostare il numero di valori da leggere per ogni singolo sensore nella #define VALUES_TO_READ_FROM_SENSOR_COUNT (tra tutti i valori a disposizione, inserire quello più alto)
 - Impostare il valore in minuti per il calcolo di un'osservazione nella #define OBSERVATIONS_MINUTES (in genere 1)
 - Impostare il numero di osservazioni necessarie a produrre un dato utile per il report nella #define STATISTICAL_DATA_COUNT (in genere 15)
4. Impostare il microcontrollore su core+1284 a 5V
5. Compilare il firmware e caricarlo attraverso il cavo USB
6. Inserire il jumper sulla board Stima I2C-Base, configurare la stazione e rimuovere il jumper a configurazione terminata

- **Modulo Stima I2C-TH**

1. Aprire lo sketch i2c-th.ino nella cartella arduino/sketchbook/rmap/i2c-th
2. Aprire il file sensors_config.h nella cartella arduino/sketchbook/libraries/RmapConfig:
 - Settare la #define USE_JSON con il valore false
 - Settare le altre #define USE_SENSOR_XXX con valori true o false a seconda dei sensori che si vuole abilitare o disabilitare
 - Impostare il numero di valori da leggere per ogni singolo sensore nella #define VALUES_TO_READ_FROM_SENSOR_COUNT (tra tutti i valori a disposizione, inserire quello più alto)
 - Impostare il valore in minuti per il calcolo di un'osservazione nella #define OBSERVATIONS_MINUTES (in genere 1 o in accordo con quanto impostato per la compilazione dello sketch rmap.ino)

- Impostare il numero di osservazioni necessarie a produrre un dato utile per il report nella `#define STATISTICAL_DATA_COUNT` (in genere 15 o in accordo con quanto impostato per la compilazione dello sketch `rmap.ino`)
- 3. Impostare il microcontrollore su core+644 a 3.3V
- 4. Compilare il firmware e caricarlo attraverso il cavo USB
- 5. Configurare il suddetto modulo attraverso lo sketch `sensor_config` presente nella cartella `arduino\sketchbook\rmap`

- **Modulo Stima I2C-Rain**

1. Aprire lo sketch `i2c-rain.ino` nella cartella `arduino/sketchbook/rmap/i2c- rain`
2. Aprire il file `sensors_config.h` nella cartella `arduino/sketchbook/libraries/RmapConfig`:
 - Settare la `#define USE_JSON` con il valore `false`
 - Settare le altre `#define USE_SENSOR_XXX` con valori `true` o `false` a seconda dei sensori che si vuole abilitare o disabilitare
 - Impostare il numero di valori da leggere per ogni singolo sensore nella `#define VALUES_TO_READ_FROM_SENSOR_COUNT` (tra tutti i valori a disposizione, inserire quello più alto)
 - Impostare la quantità di pioggia espressa in decimi di millimetro da conteggiare per ogni basculata del pluviometro nella `#define RAIN_FOR_TIP`
3. Impostare il microcontrollore su core+644 a 3.3V
4. Compilare il firmware e caricarlo attraverso il cavo USB
5. Configurare il suddetto modulo attraverso lo sketch `sensor_config` presente nella cartella `arduino\sketchbook\rmap`

- **Compilazione e caricamento del bootloader Digitecboot**

1. Aprire il file `avr_conf` nella cartella `digitecboot`
2. Editare la variabile `AVR_DIR` inserendo il percorso corretto per la directory AVR
3. Nella cartella `digitecboot` compilare il bootloader attraverso
 - `make atmega644p`
 - `make atmega1284p`
4. La compilazione produce i seguenti files:
 - `digitecboot_atmega644p.hex`: bootloader per core+644
 - `digitecboot_atmega1284p.hex`: bootloader per core+1284
5. Caricare il bootloader attraverso un programmatore impostando i seguenti parametri:
`stk500v1`, 19200 baud, Low fuse `0xFF`, Extended fuse `0xFF`, High fuse `0xD8`

- **Compilazione del firmware in formato binario**

1. Compilare il firmware per i moduli Stima secondo quanto scritto ai punti precedenti
2. Localizzare il file NOME_SKETCH.ino.elf compilato (è possibile seguire la seguente guida (<http://www.instructables.com/id/HOW-TO-GET-HEX-FILE-FROM-ARDUINO-/>))
3. Localizzare la cartella di installazione dell'IDE Arduino
4. Entrare nella cartella hardware/tools/avr/bin situata nella cartella di installazione dell'IDE Arduino
5. Attraverso una console o il prompt dei comandi, eseguire
 - `avr-objcopy -I ihex -O binary NOME_SKETCH.ino.elf FIRMWARE.BIN`

- Upload del firmware tramite micro SD-Card attraverso Digitecoboot

1. Copiare il file FIRMWARE.BIN su una micro SD-Card formattata in FAT32
2. Inserire la micro SD-Card in uno dei moduli Stima Ethernet, Stima GSM/GPRS, Stima I2C-TH o Stima-Rain
3. Alimentare il modulo e attendere 30 secondi per la fine del caricamento del firmware
4. Spegnerne il modulo e rimuovere la micro SD-Card contenente il firmware
5. Se necessario, inserire la micro SD-Card utile per il funzionamento del modulo
6. Alimentare il modulo

- **Assemblaggio stazione Stima**

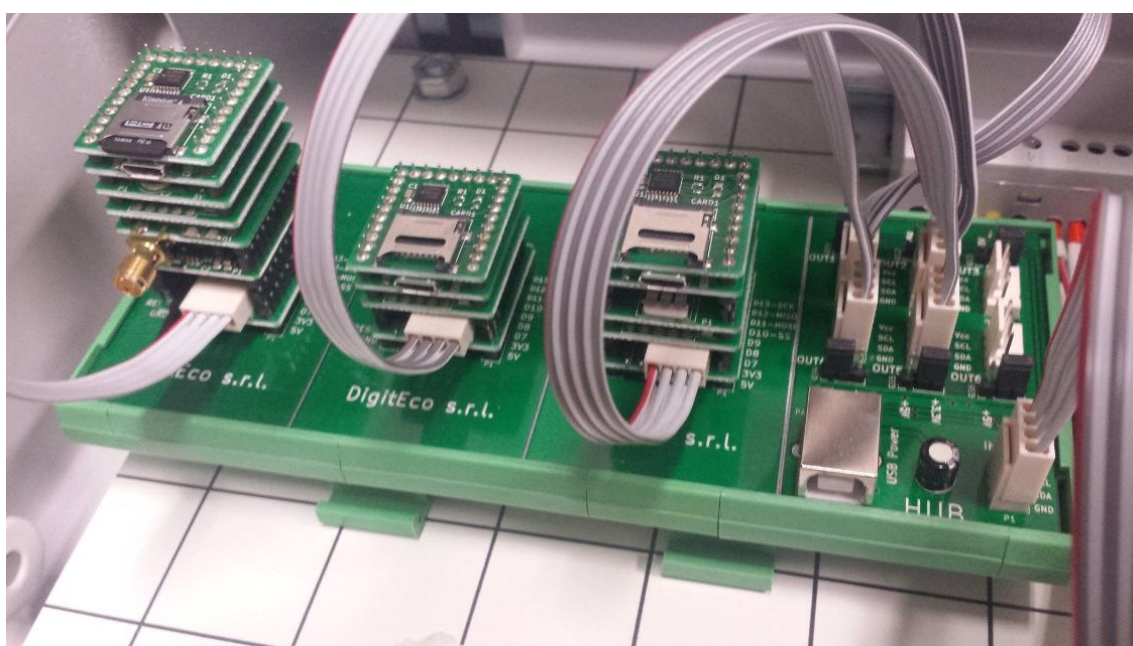
Per assemblare una stazione Stima, è necessario disporre di:

- Stima I2C-Master nella versione GSM/GPRS o Ethernet
- Stima I2C-HUB
- Stima I2C-TH
- Stima I2C-Rain
- Sensore di temperatura e umidità Digiteco TU022 (con sensore HYT221 o HYT271)
oppure sensore di temperatura ADT7420 e sensore di umidità HIH6100
- Pluviometro Digiteco PL005 o Digiteco PL010 o qualsiasi altro pluviometro con uscita a contatto
- Cavi di collegamento:
 - o per sensore di temperatura e umidità Digiteco TU022: cavo 4 poli con connettore a vite lato sensore e connettore 4 pin lato HUB
 - o o per sensori ADT7420 e HIH6100 con cavi a 4 poli per sensore a saldare lato sensore e connettore 4 pin lato HUB
 - o per pluviometro a contatto: cavo a 2 poli con connettore 4 pin lato modulo Stima I2C-Rain
- Antenna dual/quad band GPRS con connettore SMA per stazione GSM/GPRS
- Contenitore stagno IP66 310x425x160
- Batteria sigillata al piombo 12V e relativi cavi di collegamento con fusibile di protezione
- Pannello fotovoltaico 20W o sorgente di alimentazione continua 12-30 V
- DigitecoPower: se presente batteria tampone 12V con pannello fotovoltaico o sorgente di alimentazione esterna DC
- Palo acciaio inox diametro 48 mm
- Collare alluminio anodizzato per fissaggio a palo diametro 48 mm
- Morsetto alluminio anodizzato per fissaggio a sensore diametro 40 mm
- Braccetto alluminio anodizzato cilindrico per fissaggio del morsetto sensore al collare del palo
- Supporto per pluviometro
- Display I2C LCD con supporto in alluminio
- Supporti per moduli Stima su guida DIN

Una volta aver collegato i sensori, i moduli Stima, compilato i firmware e configurato tutti i moduli, così come illustrato nei capitoli precedenti, la stazione è pronta per essere utilizzata.

Di seguito un esempio di installazione di una stazione Stima GSM/GPRS con:

- Stima I2C-TH
- Stima I2C-Rain
- Stima I2C-HUB
- DigitecoPower
- Batteria sigillata 12V
- Pannello fotovoltaico



A partire da sinistra, è visibile il modulo Stima GSM/GPRS dotato di connettore tipo SMA per il collegamento dell'antenna. Al centro è visibile il modulo Stima I2C-TH e subito a destra è posizionato il modulo Stima I2C-Rain. Stima I2C-HUB collega i tre moduli, preleva l'alimentazione attraverso il connettore in ingresso in basso a destra dal modulo DigitecoPower e dispone di altre due porte libere per il collegamento di ulteriori sensori e moduli su bus I²C.

Sono ben visibili i jumper, uno per porta, per la selezione della corrispondente tensione di alimentazione (5V o 3.3V) per i moduli Stima e per i sensori I2C.



