BENVENUTI!

PROGETTO DI SPERIMENTAZIONE SULL'INTERNET DELLE COSE (IoT)

PRIMI PASSI CON MKR IoT CARRIER

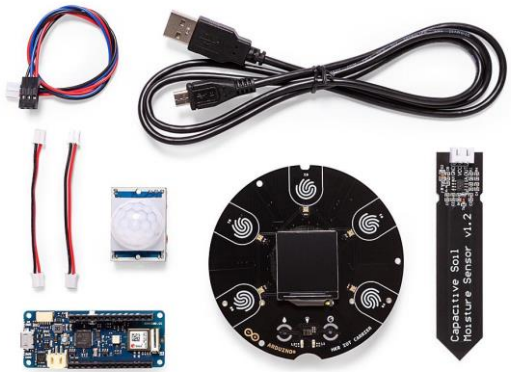
17|11|20

Con Alessia Cocco

5 APPUNTAMENTI IMPERDIBILI

- **5 tra i massimi esperti italiani** di didattica, robotica, elettronica, coding e open data
- Anche se non potrai partecipare in diretta, iscrivendoti ti assicurerai di ricevere le **videoregistrazioni** e poterne **fruire in differita** in qualsiasi momento
- **Link di iscrizione** unico a tutti gli appuntamenti

DOVE TROVARE L'EXPLORE IOT KIT



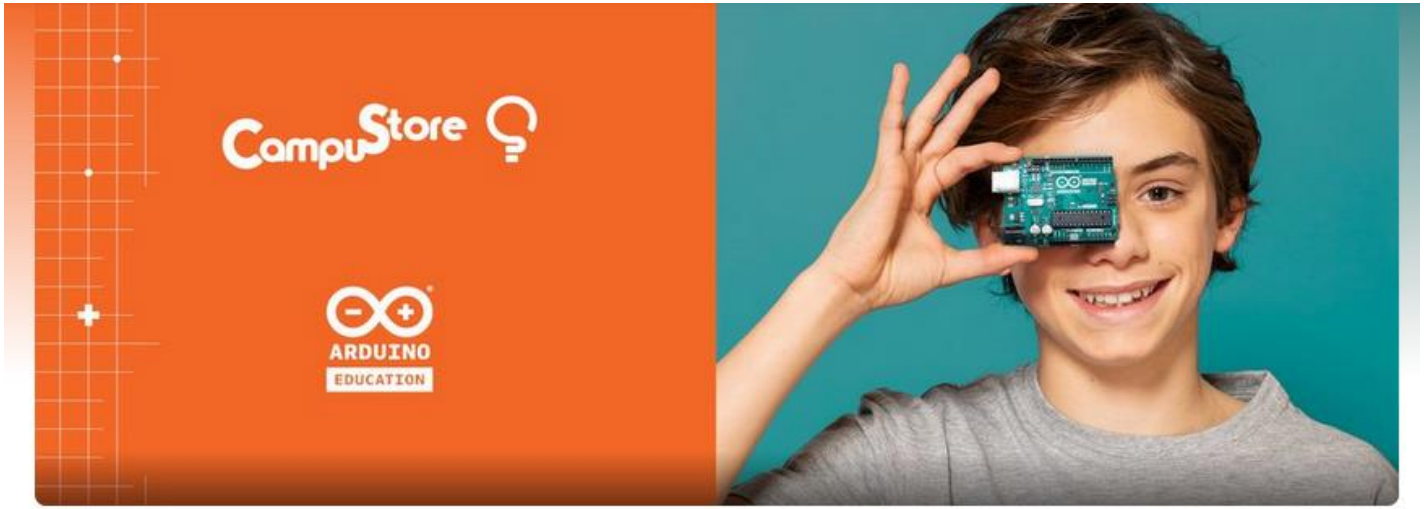
Arduino Explore IoT Kit Singolo
Codice: 333190
Codice MEPA: 333190CS

Arduino Explore IoT Kit Min. 10 pz
Codice: 334389
Codice MEPA: 334389CS

Arduino Explore IoT Kit Min. 20 pz
Codice: 334390
Codice MEPA: 334390CS

Arduino Explore IoT Kit Min. 100 pz
Codice: 334391
Codice MEPA: 334391CS

ARDUINO EDUCATION ITALIA - GRUPPO UFFICIALE FACEBOOK



Arduino Education Italia

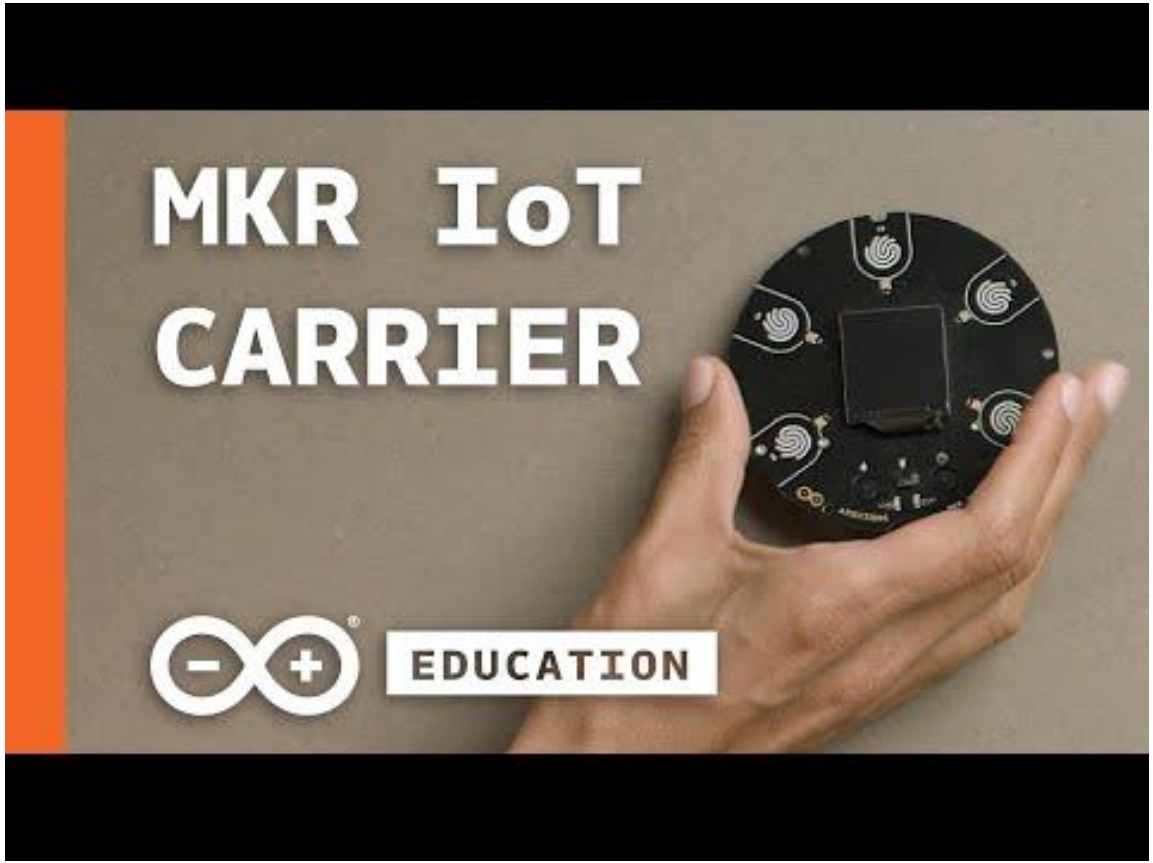
Gruppo Privato · 726 membri

Per raccogliere tutti gli educatori italiani, genitori e studenti interessati ad Arduino, CampuStore e Arduino Education hanno unito le forze e creato un **gruppo Facebook** chiamato **“Arduino Education Italia”**

PIERLUIGI VONA

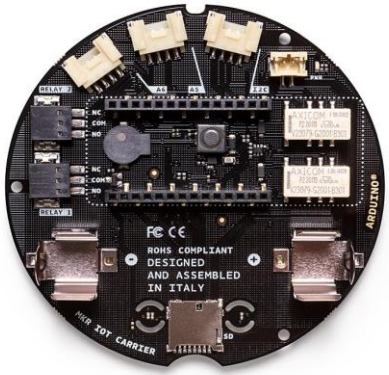
Arduino Educational Support

CHE COSA E' ARDUINO EXPLORE IoT?



MKR IOT CARRIER

L'MKR IoT Carrier può essere descritto al meglio come un'estensione della scheda MKR WiFi 1010. A differenza della scheda, il carrier non è dotato di alcun microcontrollore, il che significa che funziona solo se è collegato alla scheda Arduino.

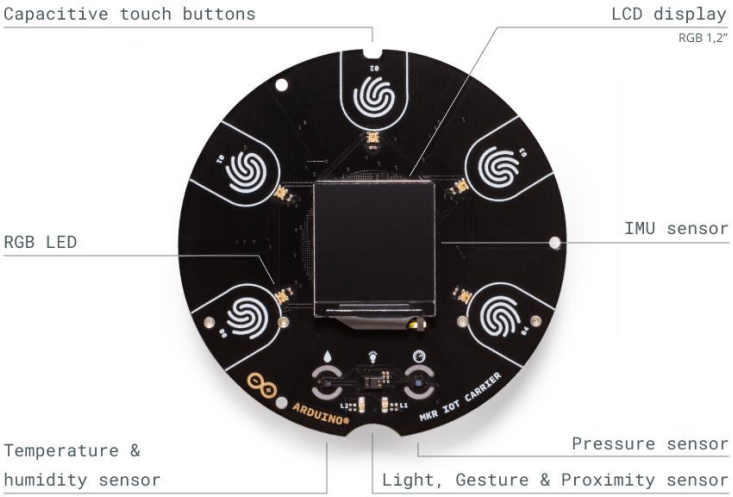


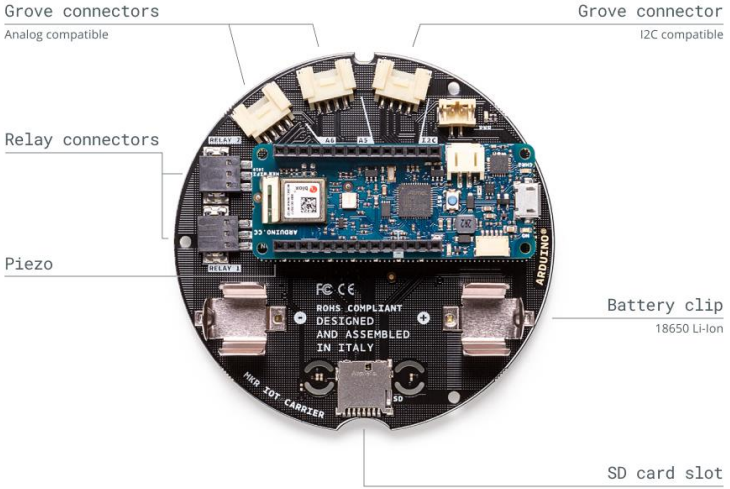
MKR IoT CARRIER

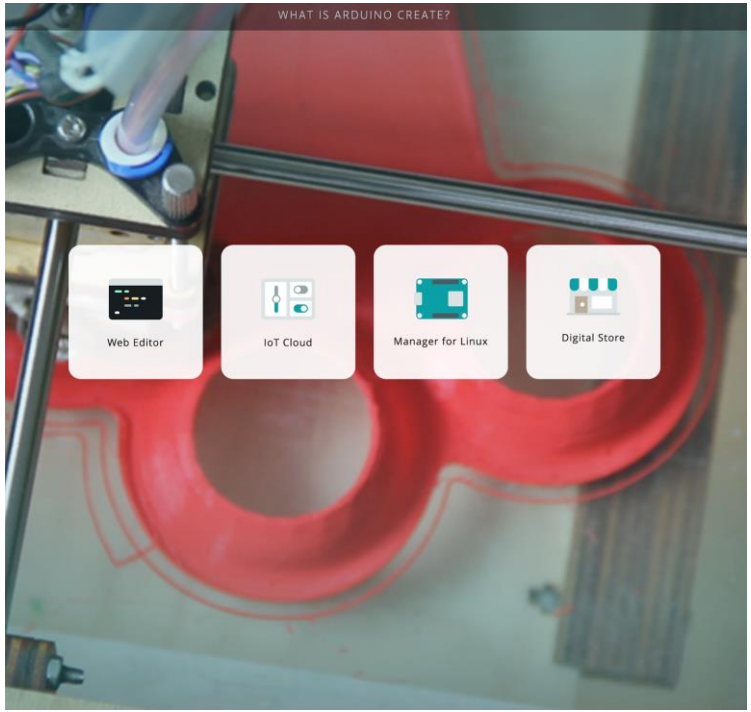
Progettato specificamente per questo kit, include:

- Due relè da 24 V
- Slot per la scheda SD
- Cinque pulsanti capacitivi
- Connettori “Plug and play” per diversi sensori
- Sensore di umidità
- Sensore di pressione
- Sensore RGBC di prossimità e riconoscimento gesti
- IMU
- Monitor RGB 1.20”
- Slot per batterie ricaricabili 18650 Li-Ion
- Cinque LED RGB

MKR IOT CARRIER



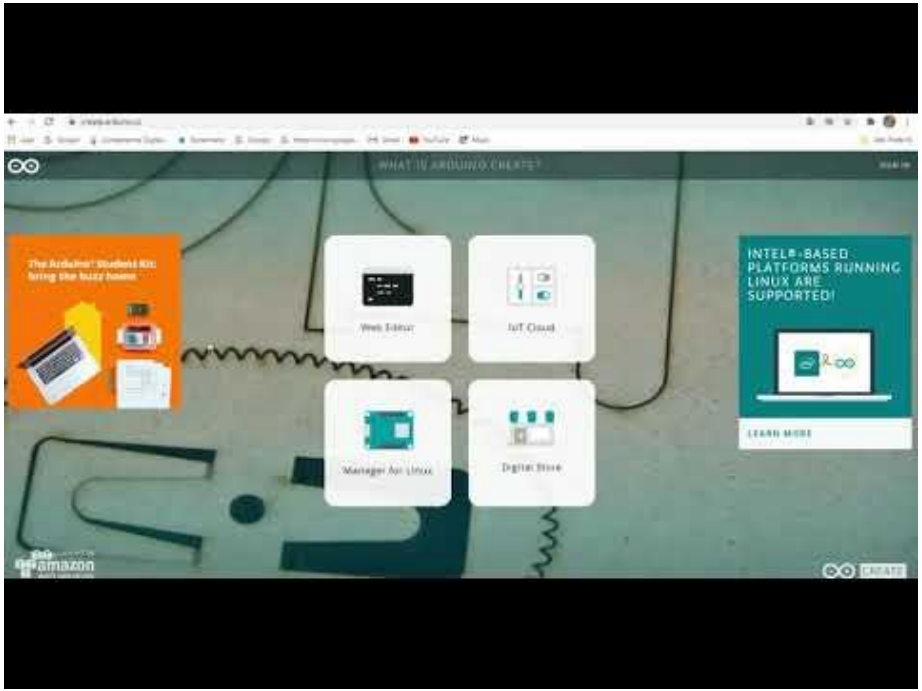




ARDUINO CREATE

Per seguire le attività del corso, dovremo effettuare il login ad Arduino Create
https://create.arduino.cc/editor
e installare il plugin Arduino Web Editor
https://github.com/arduino/arduino-create-agent
(a questo punto sulla piattaforma troveremo una guida).

Arduino Web Editor permette di scrivere codice e caricare sketch su qualsiasi scheda ufficiale di Arduino dal proprio browser web (Chrome, Firefox, Safari e Edge).



WEB EDITOR

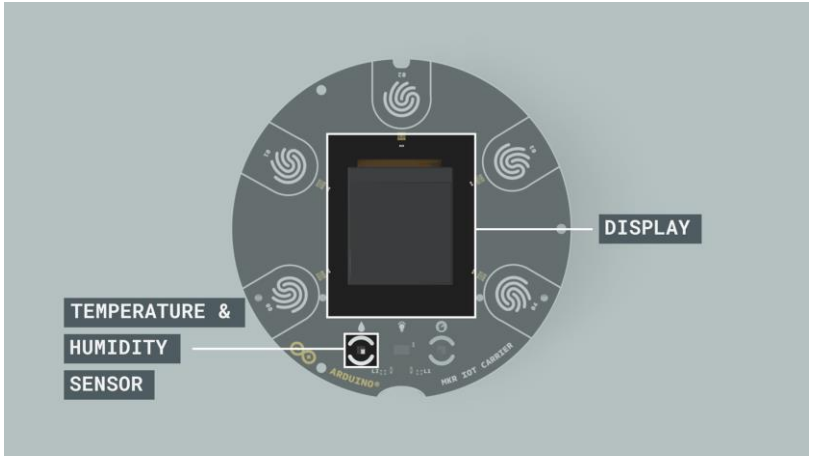
L'editor web è un vero e proprio IDE online che permetterà agli studenti di creare il loro codice passo dopo passo e poi caricarlo sulla scheda.

All'interno troverete sia le librerie che gli esempi necessari sia il monitor seriale che permette di visualizzare facilmente i dati dei sensori dell'IoT Carrier.

COME RACCOGLIERE DATI, ACCEDERE AI SENSORI DEL CARRIER

Per leggere i valori dai sensori, includeremo nel nostro sketch una libreria specifica chiamata **Arduino_MKRIoTCarrier**.
Tutti i sensori della MKR IoT Carrier sono accessibili chiamando la linea di comando **carrier.readSensor()**, dove readSensor() deve corrispondere al sensore che stiamo utilizzando.

E infine, fondamentalmente, tutto ciò che passa tra le parentesi di questa funzione può essere stampato **carrier.display.print ("Temp: ")**;

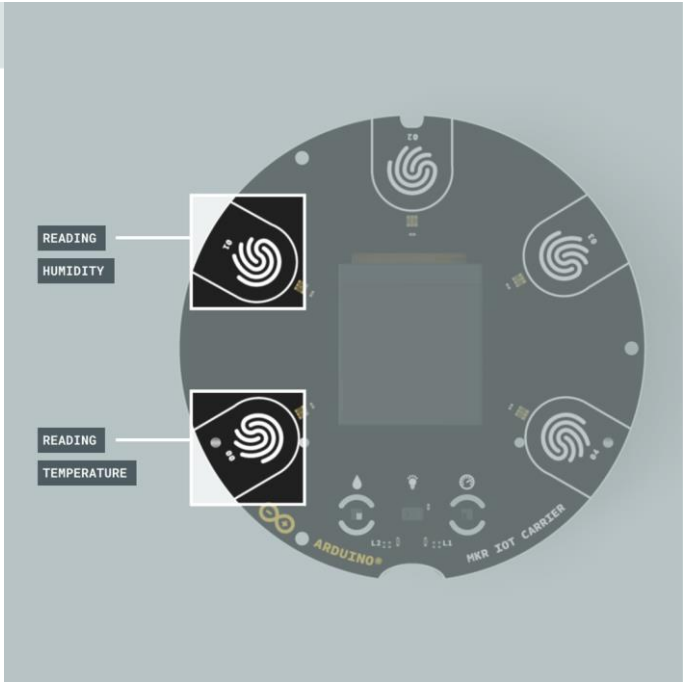


Attivital.ino | ReadMe.adoc

```
// Arduino_MKRIoTCarrier - Version: Latest
#include <Arduino_MKRIoTCarrier.h> //libreria necessaria a leggere i valori del sensore

MKRIoTCarrier carrier; //creo l'oggetto carrier

bool CARRIER_CASE = false; //indica che non stiamo usando la custodia in plastica
float temperature=0; //inizializzazione della variabile temperature
float humidity=0; //inizializzazione della variabile humidity



void setup() {
  Serial.begin(9600);//imposto la comunicazione seriale a un baudrate (velocità di bit al sec) di 9600
  while (!Serial);//se non apriamo il Serial Monitor, non succederà nulla
  carrier.begin();//inizializzazione del carrier

}
```

Attivital.ino | ReadMe.adoc

```
19 void loop() {
20   // legge i dati rilevati dai due sensori
21   temperature = carrier.Env.readTemperature();
22   humidity = carrier.Env.readHumidity();
23   //Update touch buttons
24   carrier.Buttons.update();
25   // scrive i valori letti dai due sensori
26   Serial.print("Temperature = ");
27   Serial.print(temperature);
28   Serial.println(" °C");
29
30   Serial.print("Humidity = ");
31   Serial.print(humidity);
32   Serial.println(" %");
33
34   if (carrier.Button1.onTouchDown()) {//comando che controlla il pulsante tattile
35     printTemperature();
36   }
37     if (carrier.Button2.onTouchDown()) {
38     printHumidity();
39   }
```

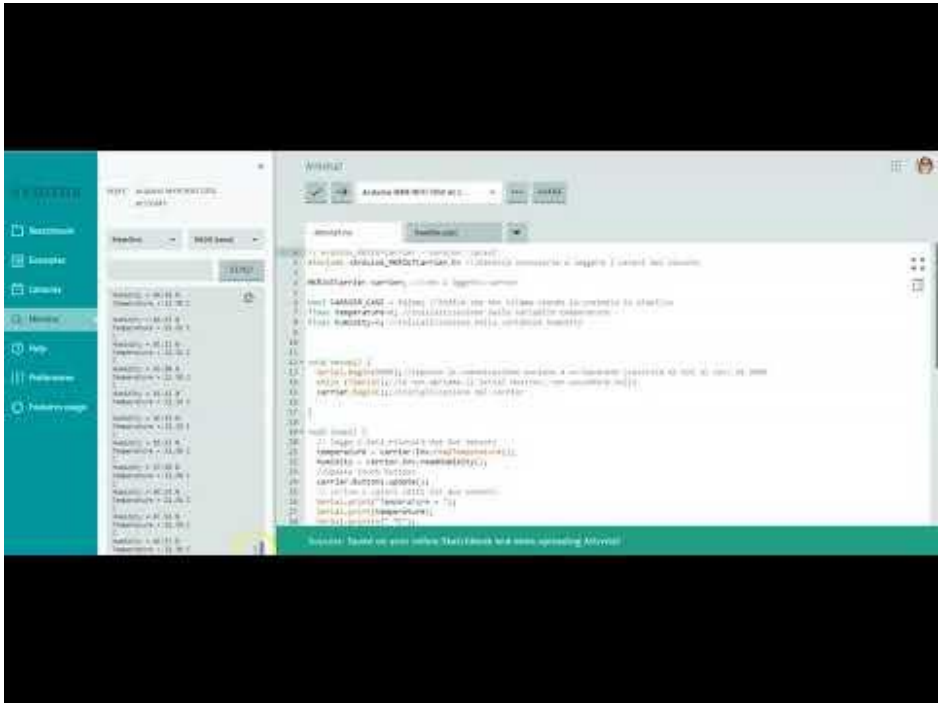


VISUALIZZARE I DATI

Attivita1.ino | ReadMe.adoc

```
41  void printTemperature() {
42  //imposta il colore dello sfondo display e del testo
43  carrier.display.fillScreen(ST77XX_RED); //sfondo rosso
44  carrier.display.setTextColor(ST77XX_WHITE); //testo bianco
45  carrier.display.setTextSize(2); //dimensione medie del testo
46
47  carrier.display.setCursor(20, 110); //coordinate x,y del punto sul display in cui inizieremo a visualizzare. E'indicata in pixel
48  carrier.display.print("Temp: ");
49  carrier.display.print(temperature);
50  carrier.display.println(" C");
51  }
52
53  void printHumidity() {
54  //imposta il colore dello sfondo display e del testo
55  carrier.display.fillScreen(ST77XX_BLUE); //sfondo blu
56  carrier.display.setTextColor(ST77XX_WHITE); //testo bianco
57  carrier.display.setTextSize(2); //dimensione medie del testo
58  carrier.display.setCursor(20, 110);
59  carrier.display.print("Humi: ");
60  carrier.display.print(humidity);
61  carrier.display.println(" %");
62  }
63
```

VISUALIZZARE I DATI



SFIDA

Proviamo a cambiare il colore dello sfondo del display, in base al livello di umidità. Possiamo farlo utilizzando un'istruzione condizionale (if/else), utilizzando le soglie sottostanti:

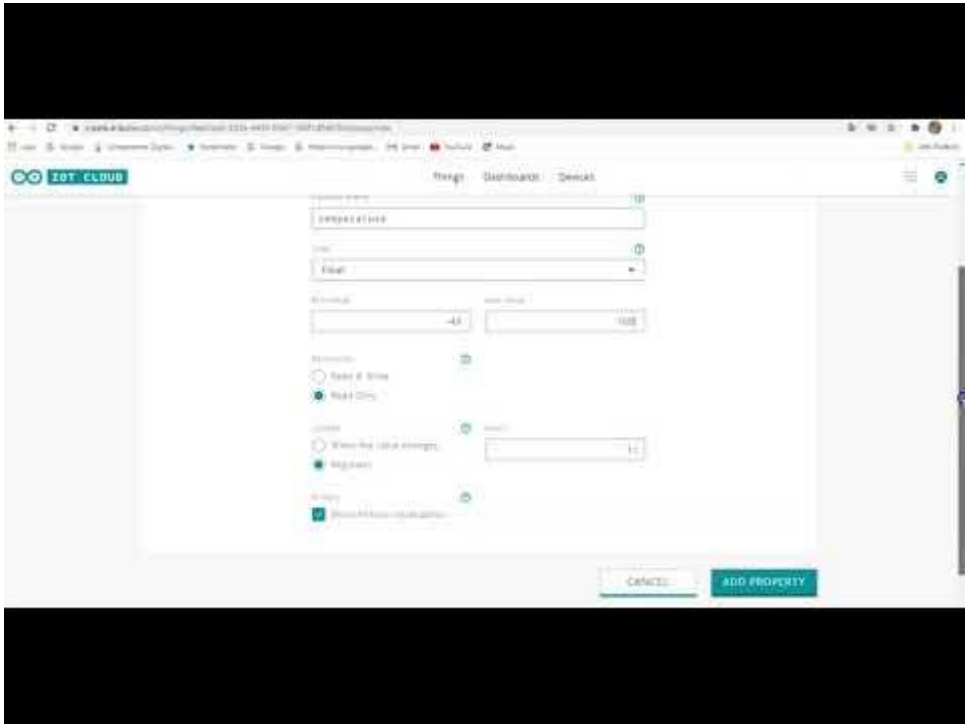
Se l'umidità è superiore all'80%, cambiare il colore dello sfondo in rosso
Se l'umidità è compresa tra il 60% e l'80%, cambiare il colore dello sfondo in giallo
Se l'umidità è compresa tra il 30% e il 60%, cambiare il colore di sfondo in blu

INVIARE I DATI NEL CLOUD

Utilizziamo Arduino IoT Cloud, dove imposteremo il nostro Arduino per inviare dati via Wi-Fi al cloud. Questo ci permetterà di vedere i dati dai nostri dispositivi in tempo reale da qualsiasi parte del mondo.

Configuriamo la nostra scheda MKR WiFi 1010 per lavorare con il cloud. Per questo processo è necessario collegare la scheda Arduino fisica all'account Create, e genererà una chiave unica che aiuterà il cloud a riconoscere la scheda.

https://create.arduino.cc/iot/



```ino
#include "thingProperties.h"
#include <Arduino_MKRIoTCarrier.h>
MKRIoTCarrier carrier;
bool CARRIER_CASE = false;

void setup() {
  // Initialize serial and wait for port to open:
  Serial.begin(9600);
  // This delay gives the chance to wait for a Serial Monitor without blocking if none is found
  delay(1500);

  // Defined in thingProperties.h

  initProperties();

```

Lo sketch generato automaticamente contiene già le variabili temperatura e umidità, quindi dobbiamo solo aggiungere alcune funzioni per accedere ai sensori sulla MKR IoT Carrier.

Le funzioni presenti in questa porzione di codice sono autogenerate e non vanno modificate, in quanto consentono la comunicazione della scheda Arduino con il Cloud e il rilevamento di eventuali errori

```
16   // Connect to Arduino IoT Cloud
17   ArduinoCloud.begin(ArduinoIoTPreferredConnection);
18
19
20  //Get Cloud Info/errors , 0 (only errors) up to 4
21   setDebugMessageLevel(2);
22   ArduinoCloud.printDebugInfo();
23
24   //Wait to get cloud connection to init the carrier
25   while(ArduinoCloud.connected() != 1){
26     ArduinoCloud.update();
27     delay(500);
28   }
29
30   delay(500);
31   carrier.begin();
32
33
34
35  }
```

```
43  void loop() {
44    ArduinoCloud.update();
45    // Your code here
46    temperature = carrier.Env.readTemperature();
47    humidity = carrier.Env.readHumidity();
48
49    Serial.print(temperature);
50    Serial.print(",");
51    Serial.println(humidity);
52
53    delay(1000);
54
55  }
56
```

Per prima cosa, abbiamo una funzione che aggiorna il cloud (non dovremmo cancellare o cambiare neanche questa). Poi, leggeremo la temperatura e l'umidità dal carrier, e le memorizzeremo nelle variabili temperature e humidity. Queste sono collegate alle proprietà che abbiamo creato in precedenza nella dashboard di Arduino IoT Cloud. L'ultima cosa che facciamo è leggere i valori nel Serial Monitor.

ACCESSO AI DATI ATTRAVERSO LA DASHBOARD DEL CLOUD

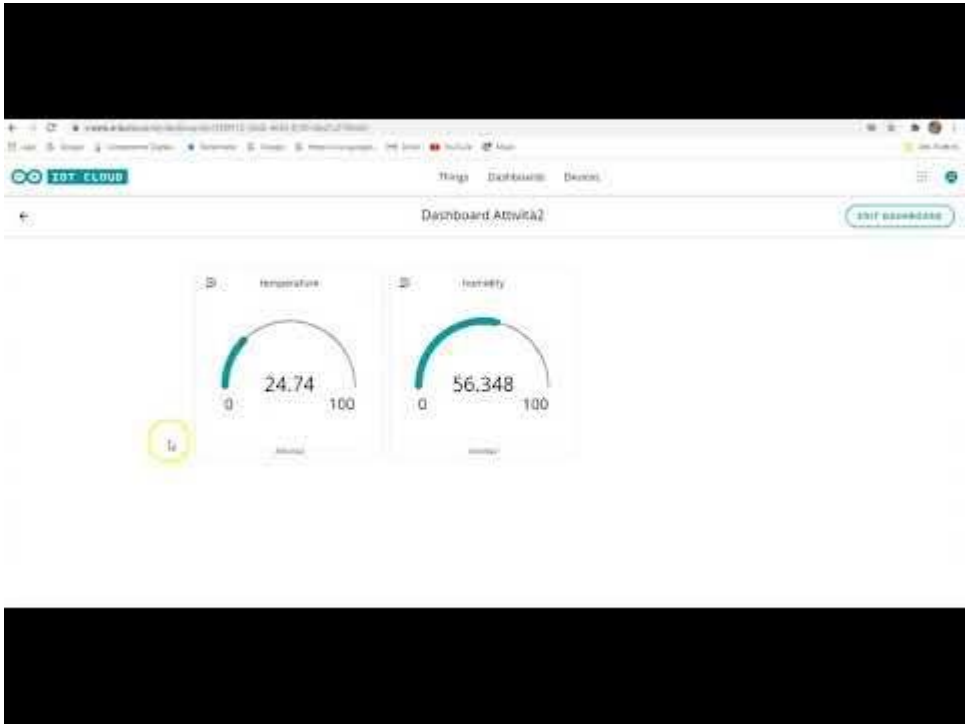
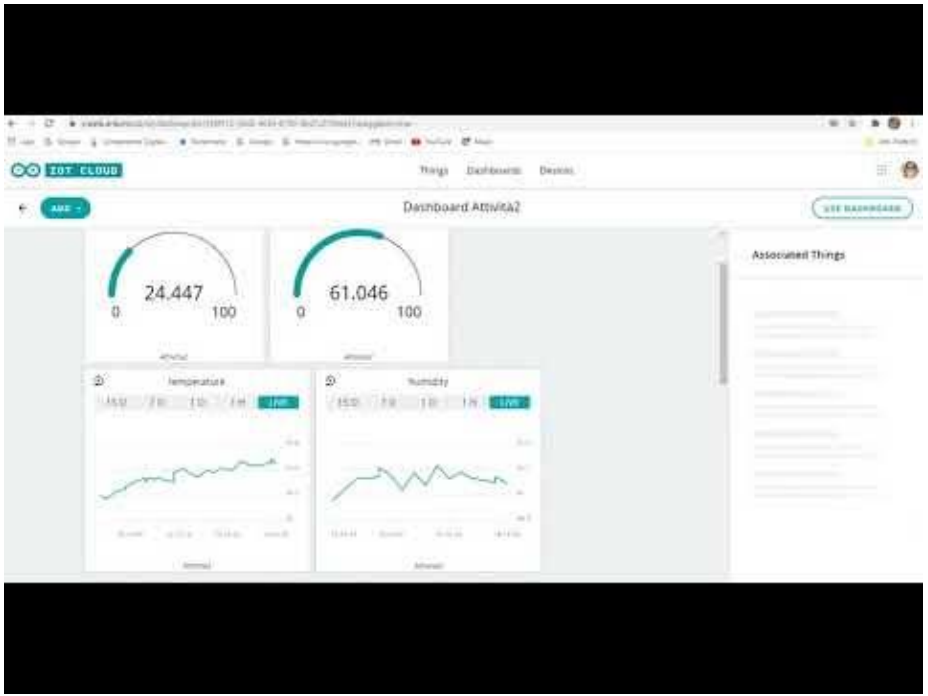


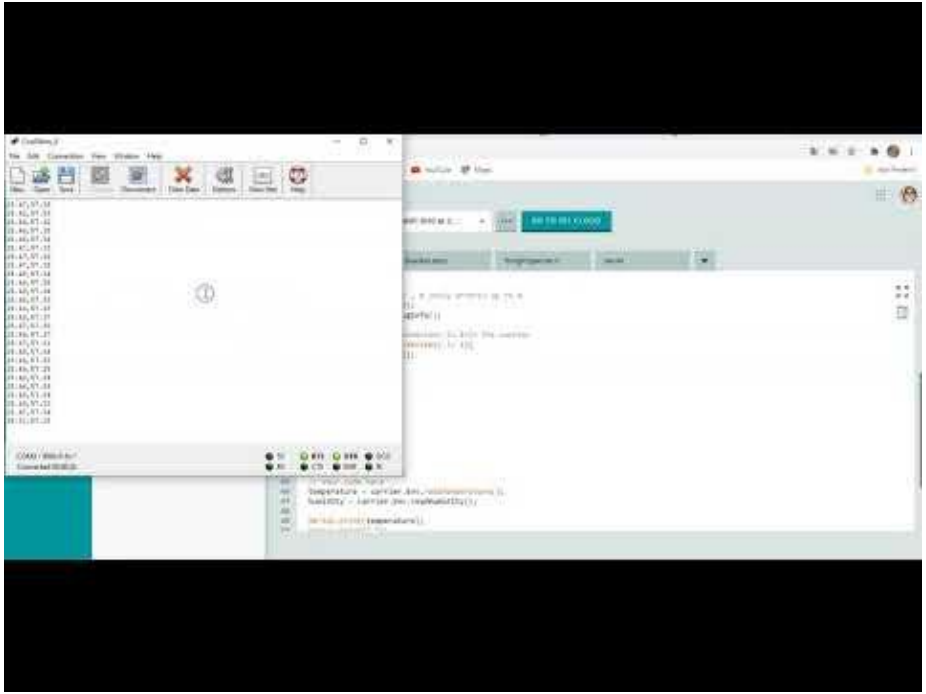
GRAFICO DEI DATI



È possibile monitorare qualsiasi cambiamento nella dashboard, che sta comunicando con la scheda tramite Wi-Fi. Questo può essere incredibilmente utile in quanto si possono monitorare i dati da qualsiasi parte del mondo, purché siano connessi a Internet. I dati registrati possono poi essere scaricati cliccando sul pulsante di download del widget.

ARCHIVIAZIONE IN UN FILE LOCALE

Un'altra tecnica per la archiviazione dei dati è lo streaming diretto dei dati in entrata dalla porta seriale in un file locale. Questo può essere utile se non abbiamo accesso al Wi-Fi.
Per questo, dobbiamo installare un monitor seriale di terze parti chiamato CoolTerm.

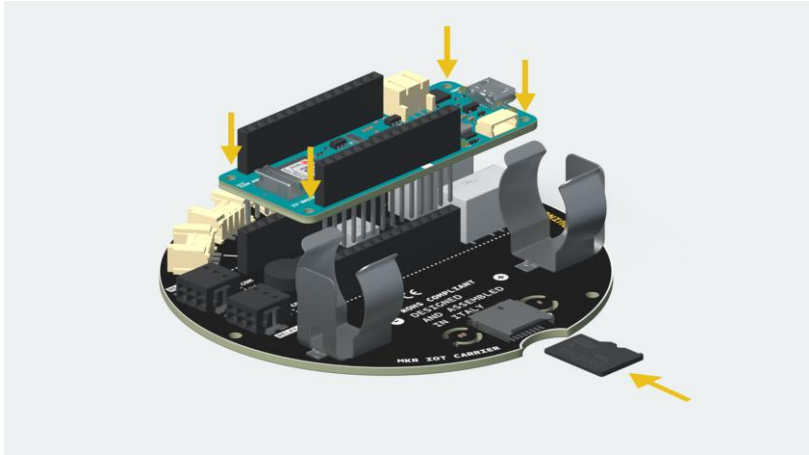


CONSERVARE & ACCEDERE AI DATI

Vedremo ora come utilizzare una scheda di **memoria SD** per memorizzare i dati in un file che potremo poi trasferire sul nostro computer e utilizzare per qualsiasi scopo.

PERCHÉ? Ci sono scenari in cui non è possibile inviare i dati in una posizione diversa via Wi-Fi o Bluetooth (come potremmo fare con il MKR Wi-Fi 1010), ma vogliamo comunque raccogliere i dati per l'analisi offline.

COME? creeremo l'oggetto dataFile, che verrà utilizzato per creare e scrivere in un file .csv.
Questo viene fatto utilizzando il comando File dataFile;



```
1  #include <Arduino_MKRIoTCarrier.h>
2  MKRIoTCarrier carrier;
3  bool CARRIER_CASE = false;
4
5  // variables
6  float temperature = 0;
7  float humidity = 0;
8
9  //track how many measurements have been made
10 int counter = 0;
11
12 // file object
13 File dataFile;
14
```

l'oggetto dataFile, verrà utilizzato per creare e scrivere in un file .csv. Questo avviene utilizzando il comando File dataFile

archiviazioneSD.ino | ReadMe.adoc

```
15 void setup() {
16   Serial.begin(9600);
17
18   while (!Serial);  //Wait to open the Serial monitor to start the program and see details on errors
19
20   //Initialize the MKRIoT carrier and output any errors in the serial monitor
21   carrier.begin();
22
23   // init SD card
24   if (!SD.begin(SD_CS)) {
25     Serial.println("Failed to initialize SD card!");
26     while (1);
27   }
28   // init the logfile
29   dataFile = SD.open("log-0000.csv", FILE_WRITE);
30   delay(1000);
31
32   // init the CSV file with headers
33   dataFile.println("temperature,humidity");
34   // close the file
35   dataFile.close();
36   delay(100);
37 }
```

Questo comando apre il file log-0000.csv e ci permette di scriverci dentro. Se il file non esiste, verrà creato automaticamente un file con quel nome

Questa riga sarà stampata all'inizio del file e ci sarà utile per la lettura dei dati

archiviazioneSD.ino | ReadMe.adoc

```
38
39 void loop() {
40   // init the logfile
41   dataFile = SD.open("log-0000.csv", FILE_WRITE);
42   delay(1000);
43  // read the sensors values
44   temperature = carrier.Env.readTemperature();
45   humidity = carrier.Env.readHumidity();
46
47   // print each of the sensor values
48   dataFile.print(temperature);
49   dataFile.print(",");
50
51   dataFile.print(humidity);
52   dataFile.println(",");
53   dataFile.close();
54
55   counter += 1;
```

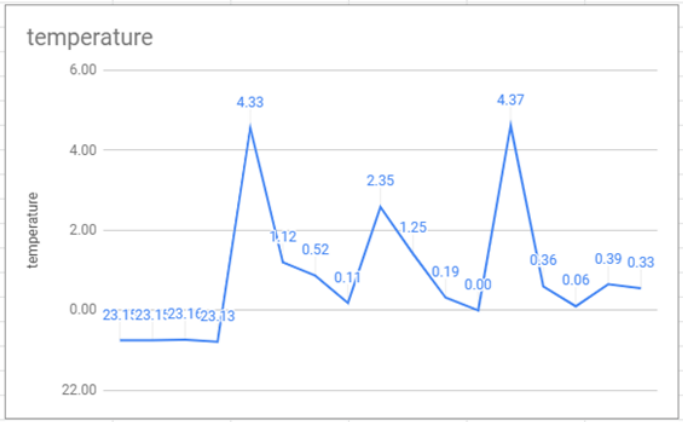
```
56    carrier.display.fillScreen(ST77XX_BLACK); //black background
57    carrier.display.setTextColor(ST77XX_WHITE); //white text
58    carrier.display.setTextSize(4); //medium sized text
59
60    carrier.display.setCursor(100, 70);
61    carrier.display.print(counter);
62    carrier.display.setCursor(20, 110);
63    carrier.display.setTextSize(2);
64    carrier.display.print("Measures taken");
65
66    // wait 1 second to print again
67    delay(1000);
68
69  }
```

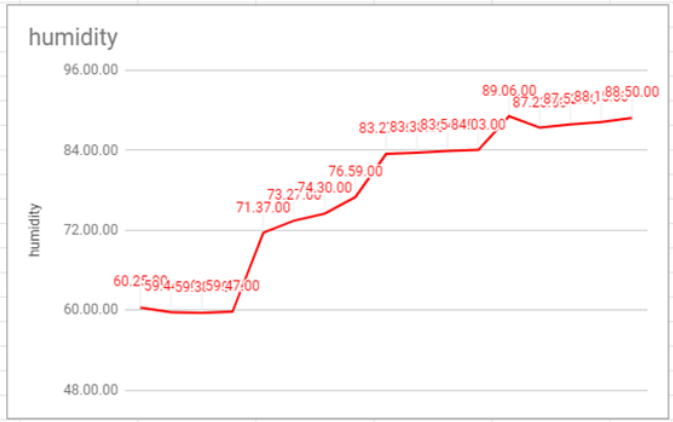
LOG-0000_bis.CSV

File Modifica Visualizza Inserisci Formato Dati Strumenti Componenti aggiuntivi Guida Appena modificato

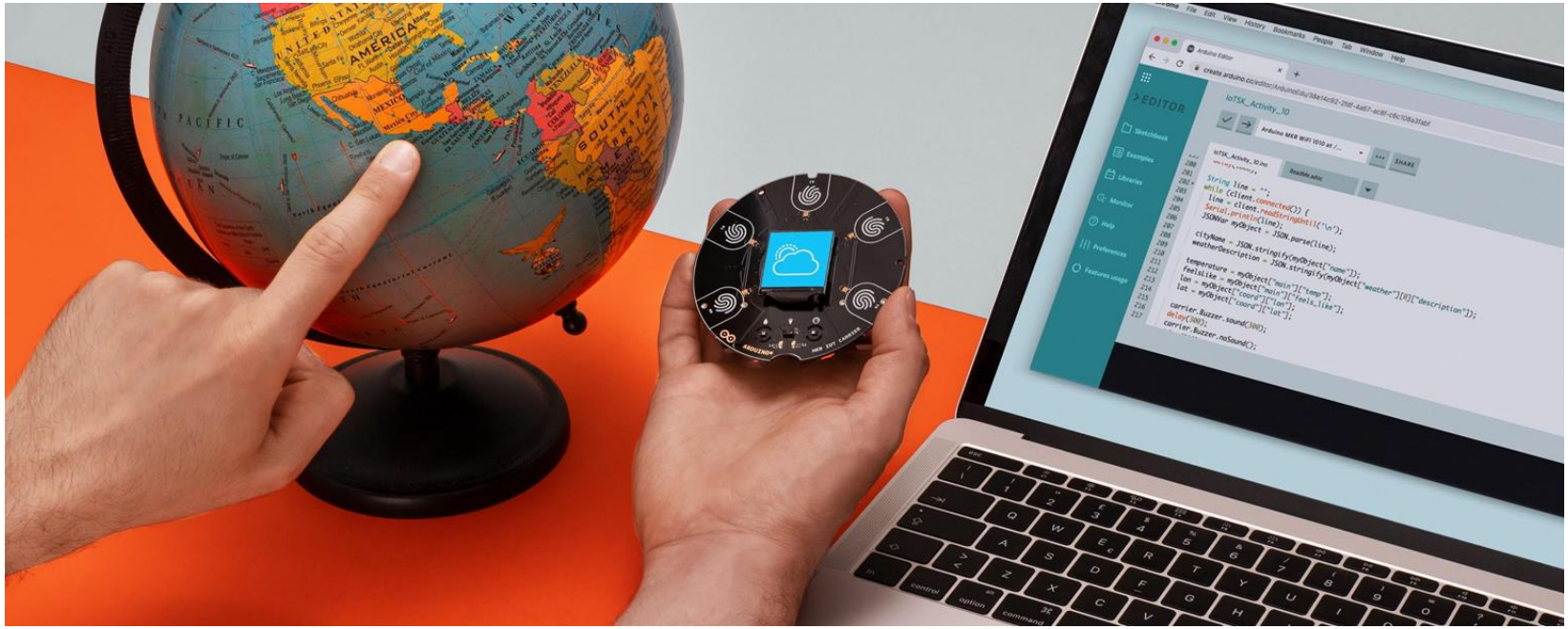
Iscritto Condividi

	A	B
1	temperature	humidity
2	23.15	60.25.00
3	23.15	59.44.00
4	23.16	59.38.00
5	23.13	59.47.00
6	28.33.00	71.37.00
7	25.12.00	73.27.00
8	24.52.00	74.30.00
9	24.11.00	76.59.00
10	26.35.00	83.27.00
11	25.25.00	83.38.00
12	24.19.00	83.54.00
13	24.00.00	84.03.00
14	28.37.00	89.06.00
15	24.36.00	87.23.00
16	24.06.00	87.52.00
17	24.39.00	88.15.00
18	24.33.00	88.50.00





LE VOSTRE DOMANDE



I PROSSIMI APPUNTAMENTI

- **Sperimentiamo con Arduino IoT Cloud** con Andrea Ferraresso - Martedì 24 novembre 2020 - 16:00-17:00
- **Sperimentiamo con il Carrier** con Domenico Aprile - Martedì 1 dicembre 2020 - 16:00-17:00
- **I primi passi con IoT** con Francesco Piersoft Paolicelli - Mercoledì 9 dicembre 2020 - 16:00-17:00

GRAZIE MILLE!

ARDUINO EXPLORE IoT KIT - SPERIMENTAZIONE DIDATTICA