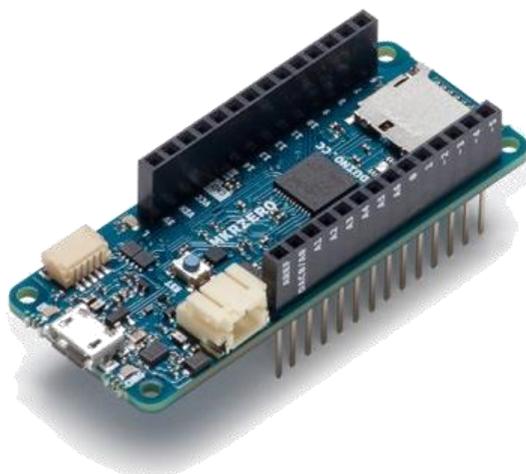


ABX00012: ARDUINO MKR ZERO



Descripción

La Arduino MKR Zero (ABX00012) es una tarjeta de desarrollo que ofrece la potencia de una tarjeta de desarrollo Arduino Zero en un formato más pequeño, establecido por el factor de forma MKR. Tiene un conector SD incorporado con interfaces SPI que le permite trabajar con archivos de audio sin hardware adicional. La tarjeta tiene un microcontrolador SAMD21, que cuenta con un núcleo ARM Cortex® M0 + de 32 bits.

Aplicaciones

La placa MKR ZERO actúa como una gran herramienta educativa para aprender sobre el desarrollo de aplicaciones de 32 bits.



Microcontrolador	SAMD21 Cortex-M0 + MCU de bajo consumo de 32 bits
Fuente de alimentación de la placa (USB / VIN)	5 V
Batería soportada (*)	Li-Po de una celda, 3.7 V, 700 mAh mínimo
Corriente de DC para Pin de 3.3V	600mA
Corriente de DC para Pin de 5V	600mA
Voltaje de funcionamiento del circuito	3.3V
Pines digitales de Entrada / Salida	22
Pines PWM	12 (0, 1, 2, 3, 4, 5, 6, 7, 8, 10, A3 - o 18 -, A4 - o 19)
UART	1
SPI	1
I2C	1
Clavijas de entrada analógica	7 (ADC 8/10/12 bit)
Pines de salida analógica	1 (DAC 10 bit)
Interrupciones externas	8 (0, 1, 4, 5, 6, 7, 8, A1 ó 16-, A2 ó 17)
Corriente DC por Pin de Entrada / Salida	7 mA
Memoria flash	256KB
Memoria Flash para Bootloader	8 KB
SRAM	32 KB
EEPROM	No
Velocidad de reloj	32.768 kHz (RTC), 48 MHz
LED integrado	32
Dispositivo USB de velocidad completa y host integrado	

¿Qué vamos a innovar hoy?

Baterías Li-Po, Pins, SD y LED de indicador.

SD

El conector SD incorporado le permite trabajar con archivos sin agregar ningún hardware adicional a la tarjeta. Además, la tarjeta SD está controlada por una interfaz SPI dedicada (SPI1).

Capacidad de la batería

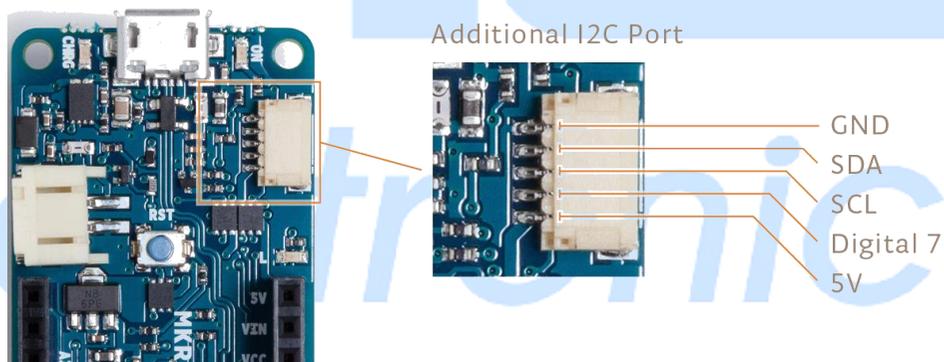
Las baterías Li-Po se cargan hasta 4,2 V con una corriente que suele ser la mitad de la capacidad nominal. Para Arduino MKR ZERO usamos un chip especializado que tiene una corriente de carga predeterminada de 350mAh. Esto significa que la capacidad MÍNIMA de la batería Li-Po debe ser de 700 mAh. Las pilas más pequeñas se dañarán con esta corriente y pueden sobrecalentarse, desarrollar gases internos y explotar, incendiando los alrededores. Le recomendamos que seleccione una batería Li-Po de al menos 700mAh de capacidad. Una pila más grande tardará más tiempo en cargarse, pero no se dañará ni se recalentará. El chip está programado con 4 horas de tiempo de carga, luego pasa al modo de reposo automático. Esto limitará la cantidad de carga a un máximo de 1400 mAh por ronda de carga.

Conector de batería

Si desea conectar una batería a su MKR Zero, asegúrese de buscar una con conector tipo JST PHR2 hembra de 2 clavijas. Polaridad: mirando a los pines del conector de la placa, la polaridad es: Izquierda = Positiva, Derecha = GND. En el MKR Zero, el conector es un tipo PH de 2 pines macho.

Puerto adicional I2C

El MKR Zero tiene un conector adicional destinado a ser una extensión del bus I2C. Es un conector de 5 clavijas de factor de forma pequeño con un paso de 1.0 mm. El puerto I2C además de las señales SDA y SCL incluye los rieles de alimentación GND y + 5V y un pin digital que puede ser útil al diseñar una expansión. El pinout se muestra en la siguiente imagen:



Vin
Este pin se puede usar para alimentar la tarjeta con una fuente regulada de 5V. Si se alimenta a través de este pin, la fuente de alimentación USB se desconecta. Esta es la única forma en que puede suministrar 5v (el rango es de 5V a un máximo de 6V). Si la placa no usa USB, este pin es una entrada.

5V

Este pin emite 5 V desde la tarjeta cuando se alimenta desde el conector USB o desde el pin VIN de la tarjeta. No está regulado y la tensión se toma directamente de las entradas. Como SALIDA, no debe usarse como un pin de entrada para alimentar la tarjeta.

VCC

Este pin emite 3.3V a través del regulador de voltaje. Este voltaje es el mismo independientemente de la fuente de alimentación utilizada (USB, Vin y batería).

LED de encendido

Este LED está conectado a la entrada de 5V desde USB o VIN. No está conectado a la energía de la batería. Esto significa que se ilumina cuando la alimentación proviene de USB o VIN, pero permanece apagado cuando la tarjeta está funcionando con la alimentación de la batería. Esto maximiza el uso de la energía almacenada en la batería. Por lo tanto, es normal que la placa funcione correctamente con la energía de la batería sin que el LED esté encendido.

LED de carga

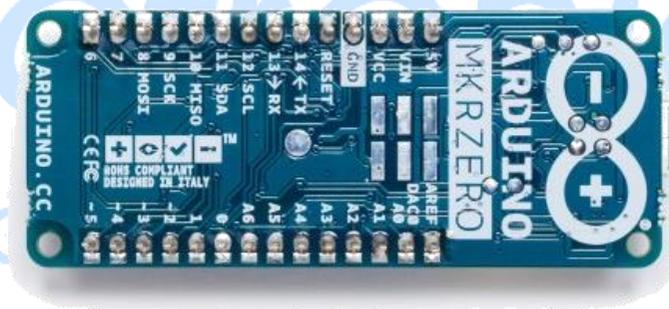
El LED de CARGA en la placa es impulsado por el chip del cargador que controla la corriente consumida por la batería Li-Po durante la carga. Por lo general, se enciende cuando la tarjeta recibe 5 V de VIN o USB y el chip comienza a cargar la batería Li-Po conectada al conector JST. Hay varias ocasiones en las que este LED comenzará a parpadear a una frecuencia de aproximadamente 2Hz. Este parpadeo se debe a las siguientes condiciones mantenidas durante mucho tiempo (de 20 a 70 minutos):

- ◆ No hay batería conectada al conector JST.
- ◆ Batería sobrecargada / dañada está conectada. No se puede recargar.
- ◆ Una batería completamente cargada pasa por otro ciclo de carga innecesario.

Esto se hace desconectando y volviendo a conectar el VIN o la propia batería mientras el VIN está conectado.

LED integrado

En MKR ZERO, el LED integrado se conecta a un pin dedicado (32) y no a 13 como en otras tarjetas. Se sugiere utilizar la definición de LED_BUILTIN.



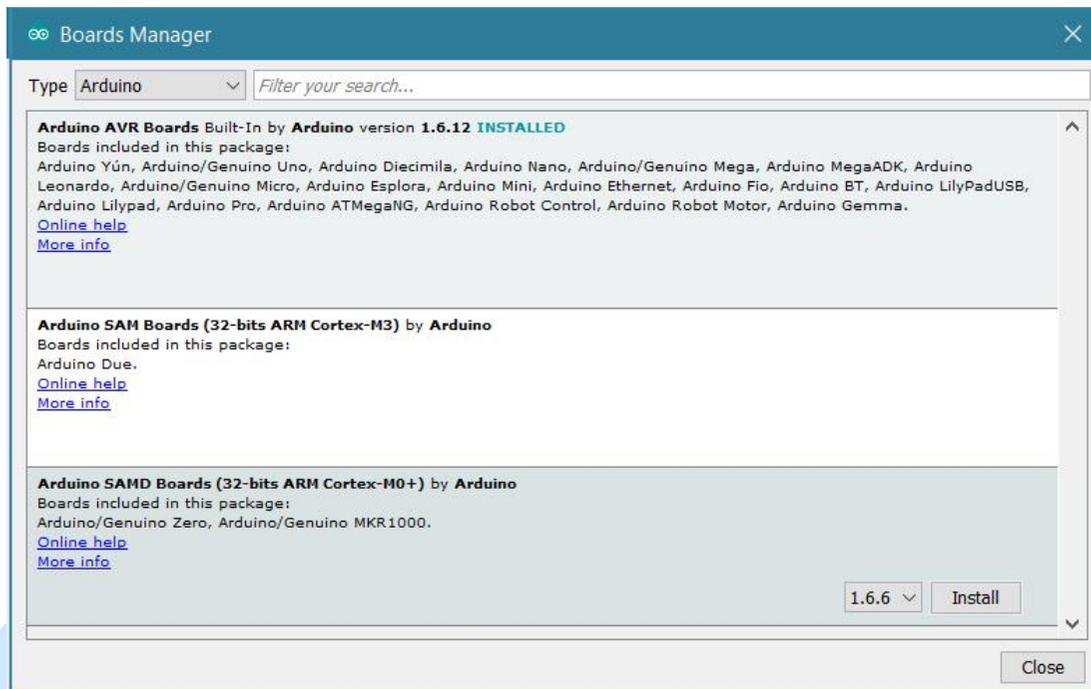
Usar Arduino MKR Zero desde el IDE de Arduino

Debe instalar el IDE de Arduino Desktop y agregar el Atmel SAMD Core.

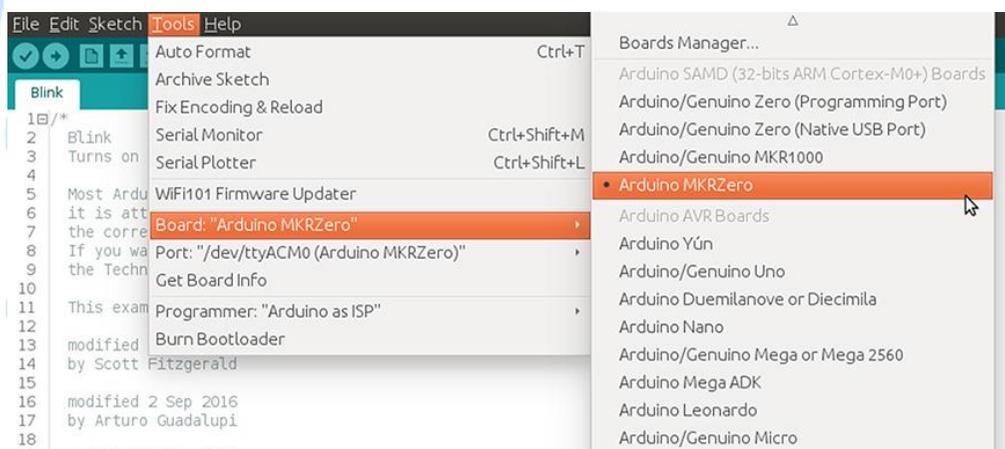
Se selecciona el menú Herramientas, después placas y al último Administrador de placa.

Aquí puedes buscar MKRZero o Zero para encontrar el núcleo. Haga clic en el correcto núcleo y después clic en el botón de instalación. En la barra inferior de la ventana puede seguir el procedimiento de descarga e instalación, incluida la instalación del controlador adecuado, que necesita el sistema operativo para utilizar la placa MKRZero.

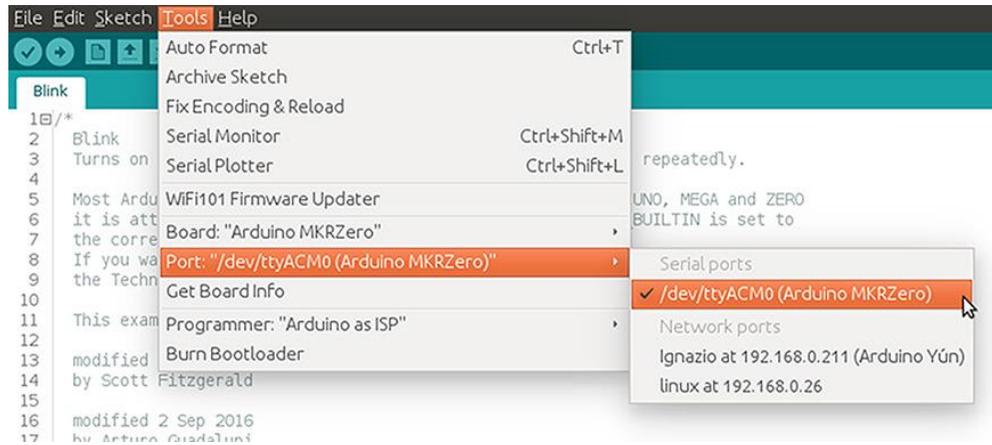
Ahora que SAMD Core está instalado, puede conectar la placa a la computadora con un cable USB estándar.



Para seleccionar tipo de placa y puerto. Desde Herramientas seleccionar la placa Arduino MKRZero.



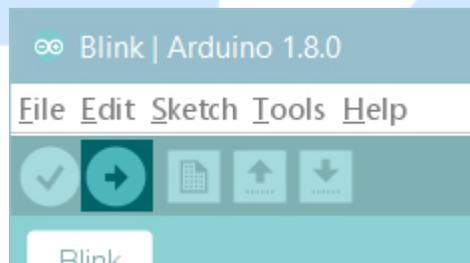
Después se selecciona el puerto que está etiquetado con el mismo nombre.



Para probar la carga de de un programa se puede descargar un ejemplo desde:

Archivo en el software Arduino (IDE), después se selecciona ejemplos; seleccione 01. Básico y luego Blink. Este programa simplemente enciende y apaga el LED incorporado al pin digital LED_BUILTIN a un ritmo de segundo.

Por ultimo presione el ícono de la segunda fila desde la izquierda en la barra superior del Software Arduino (IDE) o presione Ctrl + U y luego Cargar .



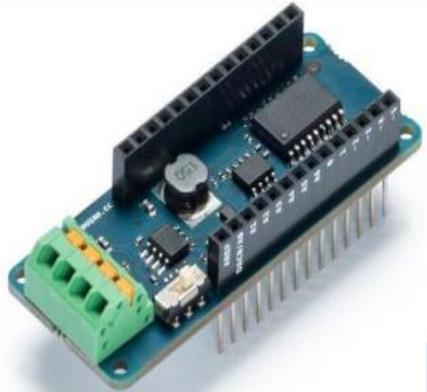
El programa será compilado y subido. Después de unos segundos, la barra inferior debería mostrar: Carga lista.

Electrónica

¿Qué vamos a innovar hoy?

A continuación, veremos algunos productos (Shields) compatibles con la tarjeta de desarrollo Arduino ABX00012. Con estos Shields/escudos podrás incrementar las capacidades y funcionalidades iniciales de la tarjeta de desarrollo, permitiéndote crear proyectos más complejos o bien con otras conectividades con las que inicialmente no se contaban, dándole solución a cualquier problema en el ámbito electrónico que se te presente.

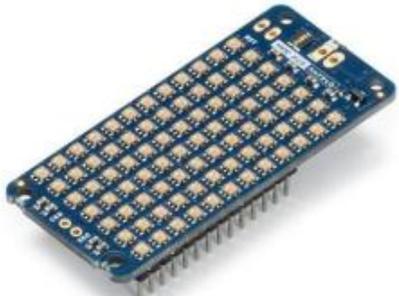
COMUNICACIONES CAN

NÚMERO DE PARTE	DESCRIPCIÓN	IMAGEN
ASX00005	Shield/escudo permite la interacción entre su Arduino MKR Board y el ecosistema CAN, que es ideal para sistemas industriales como sensores, motores, pantallas, etc; o bien aplicaciones automotrices.	

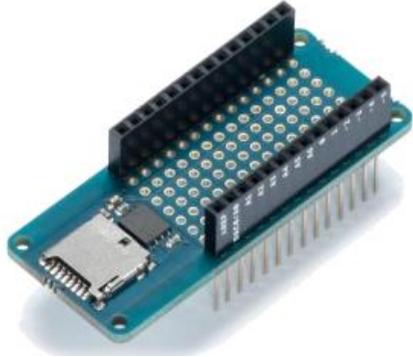
Ethernet

NUMERO DE PARTE	DESCRIPCION	IMAGEN
ASX00006	Shield/Escudo permite tener una conexión Ethernet entre su tarjeta MKR y su red, teniendo la posibilidad de conectarnos a internet. Esto es particularmente útil para dispositivos ubicados donde el ruido electromagnético es un problema o existen requisitos de seguridad especiales.	

LED

NUMERO DE PARTE	DESCRIPCION	IMAGEN
ASX00010	Shield/Escudo que contiene una matriz de 84 leds APA102 RGB organizados en 12 columnas y 7 líneas. Cada LED tiene un tamaño de 2x2 mm y el tamaño total de la matriz es de 36 x 21 mm. Cada LED se puede iluminar con un color seleccionado de una paleta de 16 millones de colores.	

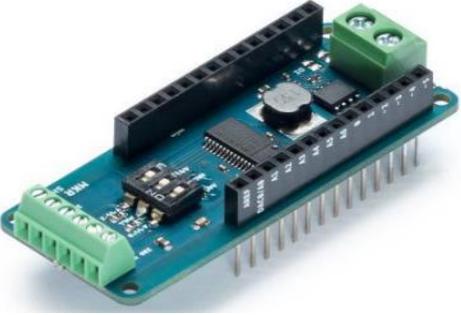
MicroSD

NUMERO DE PARTE	DESCRIPCION	IMAGEN
ASX00008	Este Shield/Escudo permite almacenamiento de datos en una tarjeta Arduino MKR a través de una microSD (no incluida). Proporciona dos megabytes de memoria flash.	

Motores

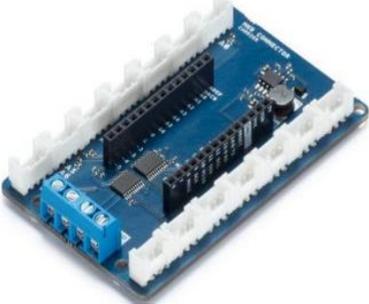
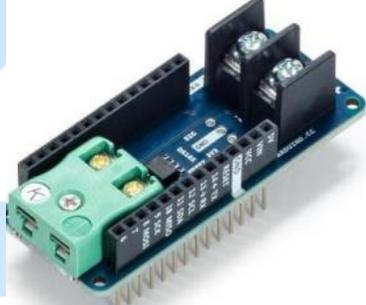
NUMERO DE PARTE	DESCRIPCION	IMAGEN
ASX00003	El ASX00003 le permite a tu Arduino MKR controlar motores de DC, servomotores y motores a pasos, ideal para desarrollar proyectos mecatrónicos, donde se requiera el control de varios motores y la lectura de varios sensores.	

RS-485

NUMERO DE PARTE	DESCRIPCION	IMAGEN
ASX00004	Shield/Escudo que permite la comunicación de tu Arduino MKR con sistemas industriales que utilizan el protocolo RS 485. como PLCs, controladores y HMI. Los sistemas industriales antiguos (por ejemplo, maquinaria, sistemas de calefacción y transportadores) pueden convertirse en dispositivos IoT a través de una conexión en serie utilizando este shield.	

¿Qué vamos a innovar hoy?

Sensores

NUMERO DE PARTE	DESCRIPCION	IMAGEN
ABX00047	Shield/escudo que permite a la tarjeta arduino MKR integrar proyectos de IoT debido a que tiene sensores integrados como: sensor de humedad y temperatura HTS221, sensor IMU LSM6DS3, sensor de presión LPS22HB, sensor de luz ambiental, gestos y proximidad APDS-9960 , 5 sensores táctiles, una pantalla tft redonda, dos relevadores, un buzzer, compartimento de microSD para almacenamiento de datos y compartimento para batería recargable 18650 Li-Ion y leds indicadores.	
ASX00007	Shield que permite le permite a las tarjetas Arduino MKR conectores Grove para conectar sensores y actuadores de forma fácil y rápida con los conectores Grove. Es ideal para proyectos escolares ayudando a una creación rápida de prototipos.	
ASX00012	Shield/escudo que permite medir la temperatura a través de un termopar tipo K y un sensor de temperatura DS18Bxx digital. Está basada en la interfaz digital del termopar MAX31855 . Ideal para proyectos escolares e industriales que busquen obtener datos de temperatura precisos y en rango extendido.	

Electrónica

¿Qué vamos a innovar hoy?

	AG Electrónica S.A.P.I. de C.V. República del Salvador N° 20 Segundo Piso Teléfono: (01)55 5130 - 7210		
ACOTACIÓN: N/A	http://www.agelectronica.com	ESCALA: N/A	REALIZO: MAUM REV:
TOLERANCIA: N/A	ARDUINO MKR ZERO		
TOLERANCIA: N/A	Fecha: 4/12/2021	No. Parte: ABX00012	



Como hacer un Datalogger midiendo la temperatura y la humedad con Arduino.

Introducción

En este artículo mostraremos como realizar un datalogger que almacenará las lecturas de temperatura y humedad dentro de una microSD para posteriormente visualizarse en el software de Excel.

Los elementos que se utilizarán son los siguientes:

- Arduino MKR Zero - [ABX00012](#)
- Carrier Arduino MKR IoT - [ABX00047](#)
- MicroSD de 16GB - [MICROSD-16GB-U3-C10](#)
- Fuente de alimentación 5V con conector microUSB - [SAW15-050-3000UD](#)

Conexión de hardware

Para iniciar con la conexión correcta del hardware siga los siguientes pasos:

1.- Coloque el Arduino MKR Zero encima del Carrier Arduino MKR IoT como se muestra en la Figura1.

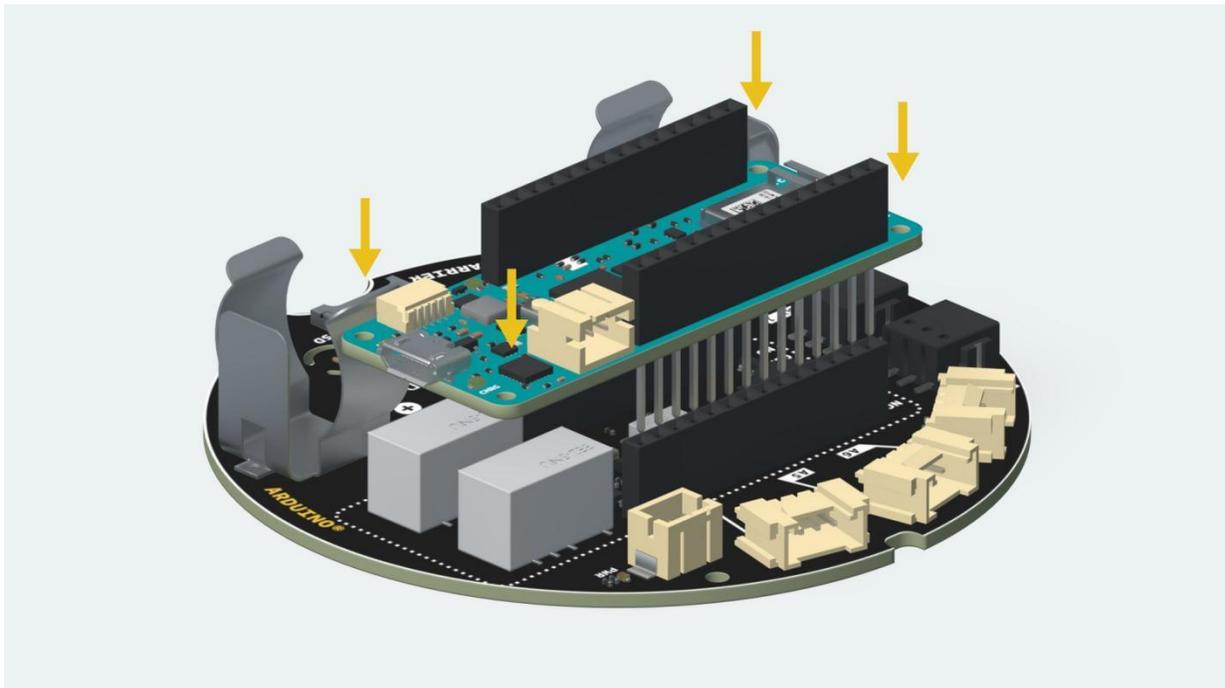


Figura1.

2.- Introduzca la microSD en el Arduino MKR Zero como se muestra en la Figura2.

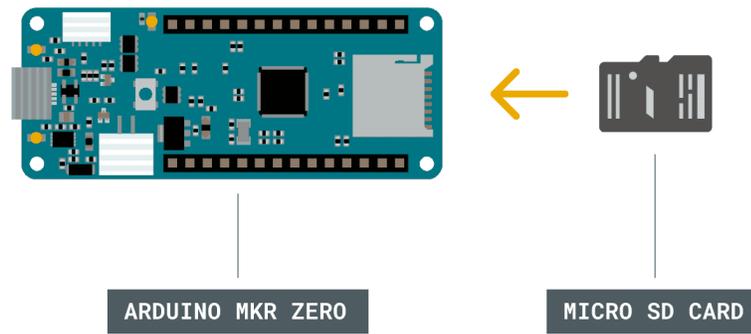


Figura2.

3. - Conecte la tarjeta Arduino MKR Zero previamente montada sobre Carrier Arduino MKR IoT a su computadora como se muestra en la Figura3.



Figura3.

Nota: Asegúrate de conectar la tarjeta de la misma forma que en la Figura1. Una buena manera de hacerlo es mirar los números de pin de la placa y de la Carrier y asegurarse de que coinciden.

Programación

A continuación veremos paso a paso la estructura del código:

I. Declaración de librerías y variables.

En este apartado declaramos las librerías y variables necesarias para el programa.

```
#include <Arduino_MKRIoTCarrier.h>
MKRIoTCarrier carrier;
const int chipSelect = 0;
float temp1=0.0;
float humidity = 0.0;
String dataString="";
```

II. Función Setup()

En la función Setup() establecemos las condiciones iniciales del programa, tales como velocidad de transmisión / recepción del puerto serial, inicialización correcta de la microSD y la inicialización correcta del Carrier Arduino MKR IoT.

```
void setup() { Serial.begin(9600);
  SD.begin(chipSelect);
  delay(5000);
  if (!carrier.begin()) {
    Serial.println("Error al inicializar los sensores");
  }
  else{
    Serial.println("Sensor iniciado correctamente!");
  }
  while (!SD.begin(chipSelect)) {
    Serial.println("Error al leer microsd");
  }
  Serial.print("Leyendo SD.....");
  Serial.println("SD preparada");
  //CARRIER_CASE = false;
}
```

III. Función void loop()

Dentro de la función void loop están todas las funciones que se ejecutaran en el programa ciclicamente.

IV. Función tempSensor()

La función voidSensor() se encargará de leer la temperatura proveniente del sensor integrado en el Carrier Arduino MKR IoT, almacenará la variable e imprimirá por serial su valor en grados Celsius.

```
void tempSensor(){
    temp1 = carrier.Env.readTemperature(CELSIUS);
    Serial.print("Temperature = ");
    Serial.print(temp1);
    Serial.println(" °C");
}
```

V. Función humiditySensor

La función humiditySensor() se encargará de leer la humedad proveniente del sensor integrado en el Carrier Arduino MKR IoT, almacenará la variable e imprimirá por serial su valor en porcentaje.

```
void humiditySensor(){
    humidity = carrier.Env.readHumidity();
    Serial.print("Humidity = ");
    Serial.print(humidity);
    Serial.println(" %");
    Serial.println();
}
```

VI. Función `SensorstorageSD()`

Esta función se encarga de almacenar en la microSD las variables que contienen las lecturas de temperatura y humedad en un formato `.csv`, formato que permite ser exportado fácilmente a Excel.

```
void storageSD(){
  dataString += "Temperatura," + String(temp1) + ",C," + "Humedad," + String(humidity) + ",%";
  File dataFile = SD.open("datalog.csv", FILE_WRITE);
  if (dataFile) {
    dataFile.println(dataString);
    dataFile.close();
  }
  else {
    Serial.println("error opening datalog.txt");
  }
}
```

Código completo

```
#include <Arduino_MKRIoTCarrier.h>
MKRIoTCarrier carrier;
const int chipSelect = 0;
float temp1=0.0;
float humidity = 0.0;
String dataString="";

void setup() {
  Serial.begin(9600);
  SD.begin(chipSelect);
  delay(5000);

  if (!carrier.begin()) {
    Serial.println("Error al inicializar los sensores");
  }
  else{
    Serial.println("Sensor iniciado correctamente!");
  }

  while (!SD.begin(chipSelect)) {
    Serial.println("Error al leer microsd");
  }

  Serial.print("Leyendo SD.....");
  Serial.println("SD preparada");
  //CARRIER_CASE = false;
}

void tempSensor(){
  temp1 = carrier.Env.readTemperature(CELSIUS);
  Serial.print("Temperature = ");
  Serial.print(temp1);
  Serial.println(" °C");
}

void humiditySensor(){
  humidity = carrier.Env.readHumidity();
  Serial.print("Humidity = ");
  Serial.print(humidity);
  Serial.println(" %");
  Serial.println();
}

void storageSD(){
  dataString += "Temperatura," + String(temp1) + ",C," + "Humedad," + String(humidity) + ",%";
  File dataFile = SD.open("datalog.csv", FILE_WRITE);
  if (dataFile) {
    dataFile.println(dataString);
    dataFile.close();
  }
  else {
    Serial.println("error opening datalog.txt");
  }
}

void loop() {
  dataString = "";
  tempSensor();
  humiditySensor();
  storageSD();
  delay(1000);
}
```