ELECTROMAGNETIC FLOWMETER INTEGRATION FOR ORIGINAL EQUIPMENT MANUFACTURERS (OEM)

ARDUINO MKR SERIES (WiFi 1010, WAN 1310, GSM 1400, NB 1500)

WITH SAMPLE CODE

7/9/2025

White Paper: Aqua Magnetics, LLC proprietary PCB with Electromagnetic Flow Sensors and Arduino MKR Series User Layer

Executive Summary

This white paper presents a modular, extensible platform combining a custom-designed printed circuit board (PCB) integrated with electromagnetic flow sensors and a user layer based on the Arduino MKR series. This two-layer architecture provides a robust foundation for developers to create tailored Internet of Things (IoT) solutions for flow measurement applications. The base layer ensures reliable sensor integration, while the Arduino MKR-based user layer offers flexible output options, including analog-to-digital conversion (ADC), I2C, Bluetooth, WiFi, LoRa, and cellular connectivity. This design empowers developers to customize functionality for diverse use cases, such as industrial fluid monitoring, environmental sensing, and smart agriculture. This document details the system architecture, integration methods, connectivity options, and potential applications, highlighting the platform's extensibility and scalability.

Target Audience: Engineers, embedded systems developers, and system integrators with expertise in the Arduino platform and associated communication protocols.

Introduction

Electromagnetic flow sensors are critical for precise measurement of conductive fluid flow, widely used in industries like water management, chemical processing, and agriculture. The custom PCB serves as a stable base layer, integrating these sensors with optimized signal conditioning and power management. The user layer, built on the Arduino MKR series, provides a familiar, open-source platform for developers to extend functionality through various communication protocols. This white paper explores the technical specifications, integration strategies, and developer extensibility of this hybrid platform, enabling innovative IoT solutions for flow monitoring.

System Architecture

The platform is designed with a two-layer architecture:

1. **Base Layer (Custom PCB with Electromagnetic Flow Sensors)**:
    - **Purpose**: Provides a reliable, optimized interface for electromagnetic flow sensors, handling signal acquisition, conditioning, and power delivery.
    - **Components**:
        - Electromagnetic flow sensors for measuring conductive fluid flow based on Faraday's Law of Electromagnetic Induction.

        - Signal conditioning circuitry (e.g., amplifiers, filters) to process raw sensor outputs.
        - Power management circuits to support low-power operation and compatibility with 3.3V Arduino MKR boards.
        - Standardized interfaces (e.g., pin headers, Grove connectors) for seamless connection to the user layer.
    - **Outputs**: Analog (e.g., 4-20mA, 0-3.3V) and digital (e.g., pulse, I2C) signals compatible with the Arduino MKR series.
2. **User Layer (Arduino MKR Series)**:
    - **Purpose**: Enables developers to extend base layer functionality with customizable processing and connectivity options.
    - **Hardware**: Arduino MKR series boards (e.g., MKR WiFi 1010, MKR WAN 1310, MKR GSM 1400, MKR NB 1500), featuring:
        - **Microcontroller**: Microchip SAMD21 Cortex-M0+ 32-bit low-power ARM MCU.
        - **Operating Voltage**: 3.3V, ensuring compatibility with the base layer.
        - **I/O**: Multiple analog/digital pins, I2C, SPI, UART, and 12-bit ADC.
        - **Connectivity**: Bluetooth (BLE), WiFi, LoRa, and cellular (GSM/NB-IoT).
        - **Eslov Connector**: Simplifies sensor integration via I2C.
        - **Power Management**: Supports Li-Po battery operation and low-power modes.
    - **Extensibility**: Open-source Arduino ecosystem with libraries, cloud integration, and community support.

This layered approach separates core sensor functionality from user-defined processing and communication, offering flexibility without compromising reliability.

Electromagnetic Flow Sensors

Electromagnetic flow sensors measure the flow rate of conductive liquids by generating a voltage proportional to fluid velocity in a magnetic field. Key features include:

- **Accuracy**: High precision for conductive fluids (e.g., water, wastewater, chemical solutions).

- **Durability**: No moving parts, reducing maintenance needs.
- **Output Signals**: Analog (4-20mA, 0-3.3V), pulse, or I2C, depending on the sensor model.
- **Applications**: Water treatment, irrigation, industrial fluid management, and environmental monitoring.

The custom PCB optimizes these sensors for integration with the Arduino MKR series, ensuring compatibility and ease of use.

Integration and Output Options

The user layer, based on the Arduino MKR series, supports multiple output protocols, allowing developers to tailor the platform to specific requirements. Below are the integration methods for each output option.

1. Analog-to-Digital Conversion (ADC)

The base layer's analog outputs (e.g., 4-20mA or 0-3.3V) connect directly to the MKR's analog input pins, which provide 12-bit ADC resolution.

- **Connection**: The custom PCB routes the sensor's analog signal to an MKR analog pin (e.g., A0). For 4-20mA signals, the PCB includes a shunt resistor or current-to-voltage converter to produce a 0-3.3V signal.
- **Programming**: Use analogRead() in the Arduino IDE to acquire and process sensor data. Calibration data from the sensor's datasheet maps ADC values to flow rates.
- **Example Use Case**: Logging water flow data in a local storage system for industrial process monitoring.

Sample Code:

```
int sensorPin = A0;

void setup() {
  Serial.begin(9600);
}

void loop() {
  int rawValue = analogRead(sensorPin); // Read analog input
  float voltage = rawValue * (3.3 / 4096.0); // Convert to voltage (12-bit ADC)
```

```
 float flowRate = (voltage / 3.3) * 100.0; // Example scaling to flow rate
 Serial.print("Flow Rate: ");
 Serial.print(flowRate);
 Serial.println(" L/min");
 delay(1000);
}
```

2. I2C Interface

The custom PCB supports I2C-compatible sensors, connecting to the MKR's I2C bus (SDA on pin 11, SCL on pin 12) or Eslov connector.

- **Connection**: The PCB routes SDA, SCL, VCC (3.3V), and GND to the MKR board. Multiple I2C devices can share the bus using unique addresses.
- **Programming**: The Wire library facilitates I2C communication. Custom or third-party libraries can simplify sensor-specific data handling.
- **Example Use Case**: Combining flow data with temperature and pressure sensors in a water treatment system.

Sample Code:

```
#include <Wire.h>
#define SENSOR_ADDRESS 0x40 // Example I2C address
void setup() {
 Wire.begin();
 Serial.begin(9600);
}
void loop() {
 Wire.beginTransmission(SENSOR_ADDRESS);
 Wire.write(0x00); // Request flow data
 Wire.endTransmission();
 Wire.requestFrom(SENSOR_ADDRESS, 2);
```

```
 if (Wire.available() >= 2) {
  int flowData = Wire.read() << 8 | Wire.read(); // Read 16-bit data
  Serial.print("Flow Data: ");
  Serial.println(flowData);
 }
 delay(1000);
}
```

3. Bluetooth (BLE)

MKR boards with BLE (e.g., MKR WiFi 1010) use the NINA-W10 module for wireless data transmission to smartphones or other BLE devices.

- **Connection**: The base layer connects to the MKR via analog or I2C, and the MKR transmits data over BLE.
- **Programming**: The ArduinoBLE library configures the MKR as a BLE peripheral, using GATT services to share flow data.
- **Example Use Case**: Real-time flow monitoring in irrigation systems, with data sent to a farmer's mobile app.

Sample Code:

```
#include <ArduinoBLE.h>

BLEService flowService("19B10000-E8F2-537E-4F6C-D104768A1214");

BLEFloatCharacteristic flowCharacteristic("19B10001-E8F2-537E-4F6C-D104768A1214", BLERead | BLENotify);

int sensorPin = A0;

void setup() {
 BLE.begin();
 BLE.setLocalName("FlowSensor");
 BLE.setAdvertisedService(flowService);
 flowService.addCharacteristic(flowCharacteristic);
```

```
 BLE.addService(flowService);
 BLE.advertise();
}
void loop() {
 BLEDevice central = BLE.central();
 if (central) {
  int rawValue = analogRead(sensorPin);
  float flowRate = (rawValue / 4096.0) * 100.0; // Example scaling
  flowCharacteristic.write(flowRate);
  delay(1000);
 }
}
```

4. WiFi (802.11 B,G,N 5GHz, 2.4 GHz)

WiFi-capable MKR boards (e.g., MKR WiFi 1010) enable cloud connectivity for real-time data logging and remote monitoring.

- **Connection**: The base layer connects via analog or I2C, and the MKR sends data over WiFi using the NINA-W10 module.
- **Programming**: The WiFiNINA library handles network connections, while HTTPClient or MQTT libraries enable cloud integration.
- **Example Use Case**: Uploading flow data from a municipal water system to a cloud dashboard.

Sample Code:

```
#include <WiFiNINA.h>
char ssid[] = "yourNetwork";
char pass[] = "yourPassword";
int sensorPin = A0;
WiFiClient client;
```

```
void setup() {
 WiFi.begin(ssid, pass);
 while (WiFi.status() != WL_CONNECTED) {
  delay(1000);
 }
 Serial.begin(9600);
}
void loop() {
 int rawValue = analogRead(sensorPin);
 float flowRate = (rawValue / 4096.0) * 100.0; // Example scaling
 if (client.connect("example.com", 80)) {
  client.print("GET /update?flow=");
  client.print(flowRate);
  client.println(" HTTP/1.1");
  client.println();
  client.stop();
 }
 delay(60000);
}
```

5. LoRa (433/868/915 MHz)

The MKR WAN 1310 supports long-range, low-power LoRa communication, ideal for remote applications.

- **Connection**: The base layer connects via analog or I2C, and the MKR transmits data to a LoRa gateway or network (e.g., The Things Network).
- **Programming**: The MKRWAN library manages LoRaWAN communication, optimizing for low power.

- **Example Use Case**: Monitoring water flow in remote agricultural fields via LoRaWAN.

Sample Code:

```
#include <MKRWAN.h>

LoRaModem modem;

int sensorPin = A0;

void setup() {
  modem.begin(EU868); // Set to 915E6 for North America
  modem.joinOTAA("appEUI", "appKey"); // Replace with your LoRaWAN credentials
  Serial.begin(9600);
}

void loop() {
  int rawValue = analogRead(sensorPin);
  int scaledValue = rawValue / 4; // Scale to fit in 1 byte
  modem.beginPacket();
  modem.write(scaledValue);
  modem.endPacket(false);
  delay(60000); // Send every minute
}
```

6. Cellular

The MKR GSM 1400 and MKR NB 1500 provide cellular connectivity (GSM or NB-IoT), enabling robust, wide-area communication for remote monitoring.

- **Connection**: The base layer connects to the MKR via analog or I2C, and the MKR transmits data over a cellular network using the onboard modem.
- **Programming**: The MKRGSM or MKRNB library manages cellular connections, allowing data transmission to cloud servers via HTTP or MQTT.

- **Example Use Case**: Remote monitoring of water flow in a city's water distribution network, with data sent to a central server via NB-IoT.

Sample Code:

```
#include <MKRNB.h>

NB nbAccess;

NBClient client;

int sensorPin = A0;

char apn[] = "yourAPN"; // Replace with your cellular provider's APN

void setup() {
 Serial.begin(9600);
 while (!nbAccess.begin(apn)) {
  Serial.println("Connecting to cellular network...");
  delay(1000);
 }
}

void loop() {
 int rawValue = analogRead(sensorPin);
 float flowRate = (rawValue / 4096.0) * 100.0; // Example scaling
 if (client.connect("example.com", 80)) {
  client.print("GET /update?flow=");
  client.print(flowRate);
  client.println(" HTTP/1.1");
  client.println("Host: example.com");
  client.println();
  client.stop();
 }
```

```
  delay(60000); // Send every minute
}
```

Developer Extensibility

The Arduino MKR-based user layer enables developers to customize and extend the platform's capabilities:

- **Custom Libraries**: Developers can create or leverage Arduino libraries to interface with specific electromagnetic flow sensors, simplifying data processing and calibration.
- **Grove Connectors**: The custom PCB may include Grove connectors for easy integration of additional sensors, enabling multi-sensor applications.
- **Cloud Integration**: The Arduino IoT Cloud or third-party platforms (e.g., AWS, Firebase) provide plug-and-play solutions for data visualization and device management.
- **Low-Power Optimization**: The MKR series supports sleep modes, allowing battery-powered systems to operate for extended periods.
- **Custom Protocols**: Developers can implement proprietary communication protocols over I2C, SPI, or UART to support unique sensor interfaces.

Use Cases

The platform supports a wide range of applications:

- **Smart Agriculture**: Monitor irrigation flow and soil moisture, transmitting data via LoRa or cellular for precision farming.
- **Water/ Wastewater Management**: Track flow rates in municipal water systems, using WiFi or cellular for cloud-based analytics.
- **Industrial Automation**: Measure fluid flow in chemical plants, integrating with PLC systems via the Iono MKR.
- **Environmental Monitoring**: Deploy battery-powered flow sensors in remote rivers or wastewater systems, using LoRaWAN or cellular for data transmission.

Challenges and Considerations

- **Voltage Compatibility**: The MKR series operates at 3.3V, requiring the custom PCB to condition sensor outputs to avoid damage.
- **Power Consumption**: Optimize sensor sampling and sleep modes for battery-powered applications to extend operational life.
- **Calibration**: Electromagnetic flow sensors require calibration, which may involve custom code or lookup tables.
- **Network Security**: Implement secure protocols (e.g., TLS for WiFi/cellular, ECC508 crypto chip) to protect data during transmission.

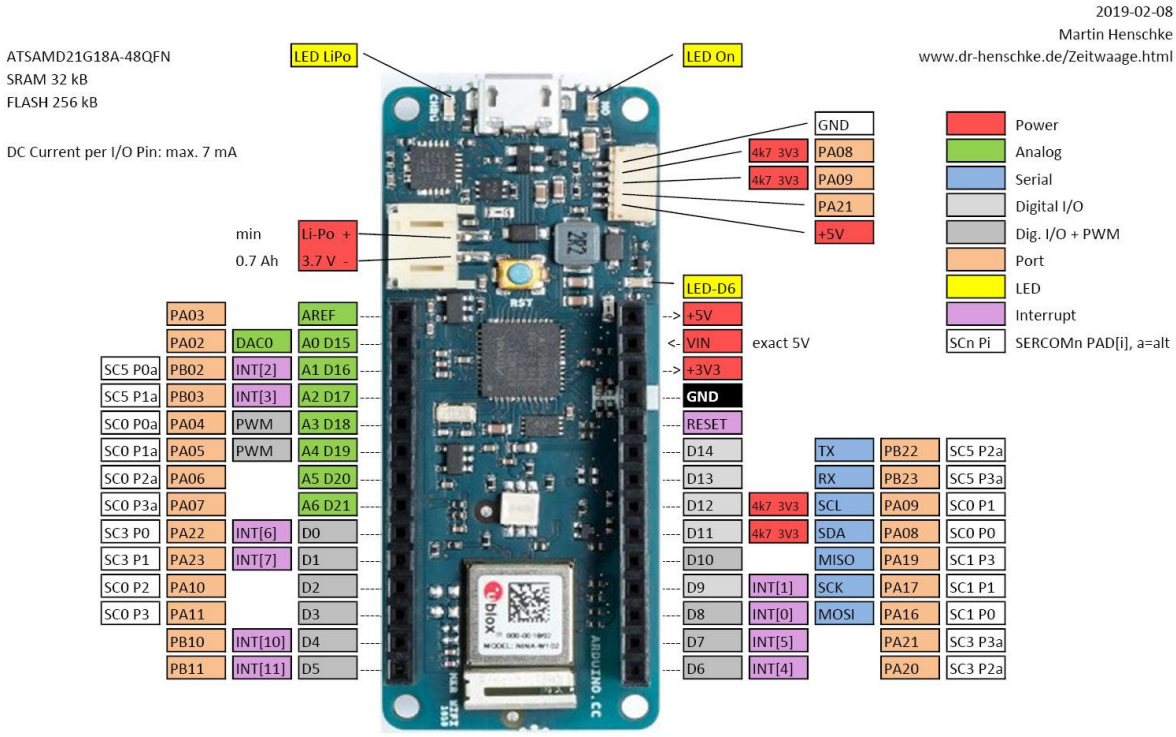
Conclusion

The combination of a custom PCB with electromagnetic flow sensors and an Arduino MKR-based user layer offers a powerful, extensible platform for IoT flow measurement solutions. The base layer ensures reliable sensor performance, while the user layer provides flexible output options (ADC, I2C, Bluetooth, WiFi, LoRa, cellular) for tailored applications. With the Arduino ecosystem's open-source tools, low-power capabilities, and robust connectivity, developers can create scalable, efficient solutions for smart agriculture, water management, industrial automation, and environmental monitoring.

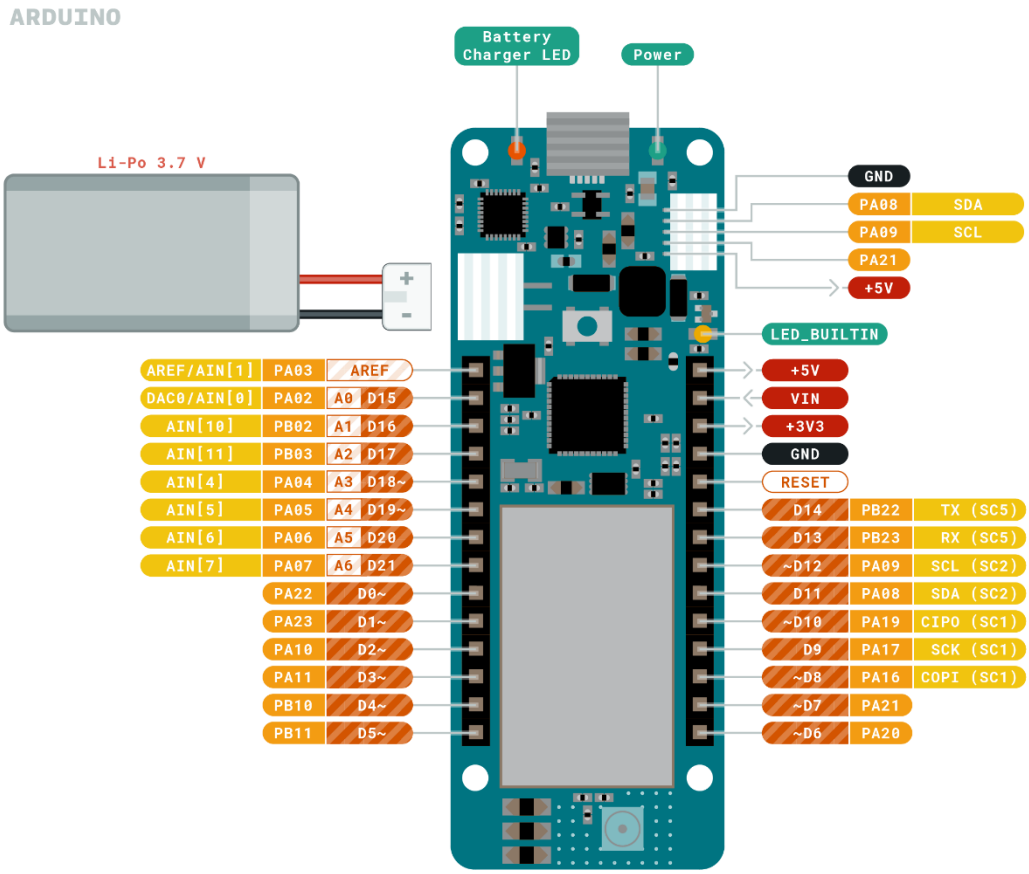
References

- Arduino Official Documentation: www.arduino.cc
- Arduino MKR Series Product Pages: store.arduino.cc
- LoRaWAN Sensor Data Tutorial: docs.arduino.cc
- MKR GSM/NB Libraries: github.com/arduino-libraries

MKR 1010 WIFI PINOUT GUIDE



MKR 1400 GSM PINOUT GUIDE



MKR NB 1500 PINOUT GUIDE