MultiFlexMeter - Gebruikershandleiding

Versie	Datum	Auteur	Wijzigingen
1.0	Feb 2026	Project Team	Initiële versie

Inhoudsopgave

---

1. Introductie

De MultiFlexMeter is een open-source IoT sensorplatform ontworpen voor het meten van rotatiesnelheid (RPM) van molens. Het systeem gebruikt LoRaWAN voor draadloze communicatie met The Things Network (TTN).

Kernfunctionaliteit

- **Pulstelling:** Meet pulsen van een tandwiel-sensor
- **RPM Berekening:** Converteert pulsen naar rotaties per minuut
- **LoRaWAN Transmissie:** Verstuurt data via EU868 frequentieband
- **Remote Configuratie:** Instellingen aanpasbaar via downlink berichten

Hardware Specificaties

Component	Specificatie
Microcontroller	ATmega1284P
Kloksnelheid	8 MHz
LoRa Radio	SX1276
Frequentieband	EU868
EEPROM	4 KB (intern)
RAM	16 KB
Flash	128 KB

---

2. Benodigdheden

Hardware

Item	Beschrijving
MultiFlexMeter PCB	Hoofdprintplaat met ATmega1284P
USBasp Programmer	ISP programmer voor firmware upload
FTDI USB-Serial Adapter	Voor debug output (optioneel)
6-pins ISP Kabel	Verbinding tussen USBasp en MFM
Sensormodule	I2C pulstellermodule (adres 0x36)

Software

Software	Versie	Download
PlatformIO Core	≥6.0	https://platformio.org/install/cli
Visual Studio Code	Latest	https://code.visualstudio.com
PlatformIO IDE Extension	Latest	VS Code Marketplace
Git	≥2.0	https://git-scm.com
AVRDUDE	≥7.0	Inbegrepen in PlatformIO

---

3. Ontwikkelomgeving Opzetten

3.1 Repository Clonen

```
git clone https://github.com/MrMisterMisterMister/Multiflexmeter
cd Multiflexmeter
```

3.2 LMIC Submodule Initialiseren

BELANGRIJK: De LoRaWAN stack (arduino-lmic) is een git submodule. Deze moet apart worden geïnitialiseerd:

```
git submodule update --init
```

Zonder deze stap zal de build mislukken met ontbrekende header-bestanden.

3.3 PlatformIO Installeren

Via pip (aanbevolen):

```
pip install platformio
```

Of via VS Code:

1. Open VS Code
2. Ga naar Extensions (Ctrl+Shift+X)
3. Zoek "PlatformIO IDE"
4. Klik Install

3.4 Project Openen

```
# Open het project in VS Code
code .
```

PlatformIO detecteert automatisch het `platformio.ini` bestand en installeert de benodigde toolchains.

---

4. Firmware Bouwen en Uploaden

4.1 Firmware Bouwen

```
pio run
```

Succesvolle output:

```
Building .pio/build/mfm_v3_m1284p/firmware.hex
=== [SUCCESS] Took X.XX seconds ===
```

De gecompileerde firmware staat in:

- `.pio/build/mfm_v3_m1284p/firmware.hex` (Intel HEX)
- `.pio/build/mfm_v3_m1284p/firmware.elf` (ELF met debug symbols)

4.2 Programmeerprotocol

De MFM wordt geprogrammeerd via **ISP (In-System Programming)**. Hierbij kan gebruik gemaakt worden van **USBasp** of **Atmel Ice**.

Parameter	Waarde
Protocol	USBasp of Atmel Ice
Interface	6-pin ISP header

Parameter	Waarde
Kloksnelheid	0.25 MHz (-B0.25 flag)
Chip Erase	Ja (-e flag)

4.3 Firmware Uploaden

Stap 1: Sluit de USBasp programmer aan

```
USBasp          MFM ISP Header
┌──────┐          ┌──────┐
│ MOSI├──────────┤ MOSI│
│ MISO├──────────┤ MISO│
│ SCK ├──────────┤ SCK │
│ RST ├──────────┤ RST │
│ VCC ├──────────┤ VCC │
│ GND ├──────────┤ GND │
└──────┘          └──────┘
```

Stap 2: Upload de firmware

```
pio run -t upload
```

Succesvolle output:

```
avrdude: AVR device initialized and ready to accept instructions
avrdude: Device signature = 0x1e9705 (ATmega1284P)
avrdude: erasing chip
avrdude: writing flash (XXXXX bytes)
avrdude: verifying flash memory
avrdude: X bytes of flash verified
=== [SUCCESS] Took X.XX seconds ===
```

4.4 Fuse Bits Configuratie

De ATmega1284P fuse bits zijn geconfigureerd in `platformio.ini`:

Fuse	Waarde	Betekenis
lfuse	0xFF	Extern kristal, full swing, geen clock divisie
hfuse	0xD1	JTAG disabled, SPI programming enabled, BOD at 2.7V
efuse	0xFF	Brown-out detection level

Fuses programmeren (kan overgeslagen worden):

```
pio run -t fuses
```

4.5 Veelvoorkomende Upload Problemen

Probleem	Oorzaak	Oplossing
`avrdude: error: could not find USB device`	USBasp niet aangesloten of driver probleem	Installeer Zadig driver (libusbK)
`avrdude: Device signature = 0x000000`	Geen verbinding met chip	Controleer ISP kabel en voeding
`avrdude: Yikes! Invalid device signature`	Verkeerde chip of klokprobleem	Verlaag ISP snelheid met `-B10`
`rc=-1`	USB permissie (Linux)	Voeg udev rules toe of gebruik `sudo`

5. EEPROM Programmeren

5.1 EEPROM Structuur

De MFM leest configuratie uit de interne EEPROM. **Zonder correcte EEPROM data start het apparaat niet!**

```
Adres   Grootte  Veld                    Beschrijving
──────────────────────────────────────────────────────────────
0x00    4        MAGIC                   "MFM\0" (validatie)
0x04    2        HW_VERSION              Hardware versie (MSB:LSB)
0x06    8        APP_EUI                 LoRaWAN Application EUI
0x0E    8        DEV_EUI                 LoRaWAN Device EUI
0x16    16       APP_KEY                 LoRaWAN Application Key
0x26    2        MEASUREMENT_INTERVAL    Meet interval in seconden
0x28    1        USE_TTN_FAIR_USE        TTN fair use policy aan/uit
0x29    1        WHEEL_TEETH_COUNT       Aantal tanden op meetwiel
──────────────────────────────────────────────────────────────
Totaal: 42 bytes
```

5.2 EEPROM Programmeer Script

Maak een aparte Arduino-sketch om de EEPROM te programmeren:

```
// eeprom_programmer.ino
// Upload dit EENMALIG naar de MFM om credentials in te stellen

#include <EEPROM.h>

struct __attribute__((packed)) rom_conf_t {
```

```
  uint8_t MAGIC[4];
  struct { uint8_t MSB; uint8_t LSB; } HW_VERSION;
  uint8_t APP_EUI[8];
  uint8_t DEV_EUI[8];
  uint8_t APP_KEY[16];
  uint16_t MEASUREMENT_INTERVAL;
  uint8_t USE_TTN_FAIR_USE_POLICY;
  uint8_t WHEEL_TEETH_COUNT;
};

void setup() {
  rom_conf_t config;

  // Magic bytes (verplicht)
  config.MAGIC[0] = 'M';
  config.MAGIC[1] = 'F';
  config.MAGIC[2] = 'M';
  config.MAGIC[3] = '\0';

  // Hardware versie (bijv. 3.0)
  config.HW_VERSION.MSB = 0x00;  // versie encoding
  config.HW_VERSION.LSB = 0xC0;  // 3.0.0 = 0x00C0

  // LORAWAN CREDENTIALS - Vervang met eigen waarden

  // APP_EUI (8 bytes, LSB first zoals in TTN console)
  // TTN toont: 70B3D57ED0XXXXXX
  // Schrijf LSB first: XX XX XX D0 7E D5 B3 70
  config.APP_EUI[0] = 0xXX;  // LSB
  config.APP_EUI[1] = 0xXX;
  config.APP_EUI[2] = 0xXX;
  config.APP_EUI[3] = 0xD0;
  config.APP_EUI[4] = 0x7E;
  config.APP_EUI[5] = 0xD5;
  config.APP_EUI[6] = 0xB3;
  config.APP_EUI[7] = 0x70;  // MSB

  // DEV_EUI (8 bytes, LSB first)
  // TTN toont: 70B3D57ED0YYYYYY
  config.DEV_EUI[0] = 0xYY;  // LSB
  config.DEV_EUI[1] = 0xYY;
  config.DEV_EUI[2] = 0xYY;
  config.DEV_EUI[3] = 0xD0;
  config.DEV_EUI[4] = 0x7E;
  config.DEV_EUI[5] = 0xD5;
  config.DEV_EUI[6] = 0xB3;
  config.DEV_EUI[7] = 0x70;  // MSB

  // APP_KEY (16 bytes, MSB first zoals in TTN console)
  // TTN toont: 2B7E151628AED2A6ABF7158809CF4F3C
  config.APP_KEY[0]  = 0x2B;  // MSB
  config.APP_KEY[1]  = 0x7E;
  config.APP_KEY[2]  = 0x15;
  config.APP_KEY[3]  = 0x16;
```

```
  config.APP_KEY[4]  = 0x28;
  config.APP_KEY[5]  = 0xAE;
  config.APP_KEY[6]  = 0xD2;
  config.APP_KEY[7]  = 0xA6;
  config.APP_KEY[8]  = 0xAB;
  config.APP_KEY[9]  = 0xF7;
  config.APP_KEY[10] = 0x15;
  config.APP_KEY[11] = 0x88;
  config.APP_KEY[12] = 0x09;
  config.APP_KEY[13] = 0xCF;
  config.APP_KEY[14] = 0x4F;
  config.APP_KEY[15] = 0x3C;  // LSB

  // CONFIGURATIE INSTELLINGEN

  // Meetinterval (300 = 5 minuten)
  config.MEASUREMENT_INTERVAL = 300;

  // TTN Fair Use Policy (1 = aan, 0 = uit)
  config.USE_TTN_FAIR_USE_POLICY = 1;

  // Aantal tanden op meetwiel (standaard 91)
  config.WHEEL_TEETH_COUNT = 91;

  // Schrijf naar EEPROM
  EEPROM.put(0, config);

  // Verificatie LED
  pinMode(LED_BUILTIN, OUTPUT);
  digitalWrite(LED_BUILTIN, HIGH);
}

void loop() {}
```

5.3 EEPROM Uitlezen (Debug)

Om de huidige EEPROM-waarden te bekijken, gebruik dit script:

```
// eeprom_reader.ino
#include <EEPROM.h>

struct __attribute__((packed)) rom_conf_t {
  uint8_t MAGIC[4];
  struct { uint8_t MSB; uint8_t LSB; } HW_VERSION;
  uint8_t APP_EUI[8];
  uint8_t DEV_EUI[8];
  uint8_t APP_KEY[16];
  uint16_t MEASUREMENT_INTERVAL;
  uint8_t USE_TTN_FAIR_USE_POLICY;
  uint8_t WHEEL_TEETH_COUNT;
};
```

```
void printHex(uint8_t* data, uint8_t len) {
  for (int i = 0; i < len; i++) {
    if (data[i] < 0x10) Serial.print("0");
    Serial.print(data[i], HEX);
    if (i < len - 1) Serial.print(" ");
  }
}

void setup() {
  Serial.begin(115200);
  while (!Serial);

  rom_conf_t config;
  EEPROM.get(0, config);

  Serial.print("MAGIC: ");
  Serial.write(config.MAGIC, 3);
  Serial.print(" (valid: ");
  Serial.print(strcmp((char*)config.MAGIC, "MFM") == 0 ? "YES" : "NO");
  Serial.println(")");

  Serial.print("HW Version: ");
  uint16_t v = config.HW_VERSION.MSB << 8 | config.HW_VERSION.LSB;
  Serial.print((v >> 10) & 0x1F);
  Serial.print(".");
  Serial.print((v >> 5) & 0x1F);
  Serial.print(".");
  Serial.println(v & 0x1F);

  Serial.print("APP_EUI: ");
  printHex(config.APP_EUI, 8);
  Serial.println(" (LSB first)");

  Serial.print("DEV_EUI: ");
  printHex(config.DEV_EUI, 8);
  Serial.println(" (LSB first)");

  Serial.print("APP_KEY: ");
  printHex(config.APP_KEY, 16);
  Serial.println(" (MSB first)");

  Serial.print("Interval: ");
  Serial.print(config.MEASUREMENT_INTERVAL);
  Serial.println(" sec");

  Serial.print("TTN FUP: ");
  Serial.println(config.USE_TTN_FAIR_USE_POLICY ? "ON" : "OFF");

  Serial.print("Teeth: ");
  Serial.println(config.WHEEL_TEETH_COUNT);
}

void loop() {}
```

6. LoRaWAN Netwerk Registratie (TTN)

6.1 The Things Network (TTN) Registratie

Stap 1: Account aanmaken

1. Ga naar https://console.cloud.thethings.network
2. Selecteer de juiste cluster (Europe: `eu1.cloud.thethings.network`)
3. Maak een account aan of log in

Stap 2: Application aanmaken

1. Klik op "Create application"
2. Vul een Application ID in (bijv. `molen-monitoring`)
3. Klik "Create application"

Stap 3: Device registreren

1. Ga naar je application
2. Klik "Register end device"
3. Selecteer "Enter end device specifics manually"
4. Configureer:

Veld	Waarde
Frequency plan	Europe 863-870 MHz (SF9 for RX2)
LoRaWAN version	LoRaWAN Specification 1.0.3
Regional Parameters	RP001 Regional Parameters 1.0.3 revision A

Stap 4: Device EUIs genereren

- **JoinEUI (AppEUI):** Laat TTN genereren
- **DevEUI:** Gebruik device addres van de MFM
- **AppKey:** Klik "Generate" voor een random key

6.2 Byte Order Conversie

BELANGRIJK: De LMIC library verwacht een specifieke byte-volgorde!

Key	TTN Console Formaat	EEPROM Volgorde
APP_EUI	MSB first (70B3...)	**LSB first** (omkeren!)
DEV_EUI	MSB first (70B3...)	**LSB first** (omkeren!)
APP_KEY	MSB first	**MSB first** (niet omkeren)

Voorbeeld conversie:

```
TTN toont DEV_EUI: 70 B3 D5 7E D0 06 12 34

EEPROM (LSB first): 34 12 06 D0 7E D5 B3 70
                    ^^ byte 0          ^^ byte 7
```

6.3 Payload Formatter

Voeg de payload decoder toe aan TTN in payload formatters > Uplink > Custom JavaScript formatter. De payload is te vinden op:

```
https://gist.github.com/MrMisterMisterMister/6380a30f83c20c0d3bc922c4275ac2a4
```

---

7. Over-The-Air (OTA) Configuratie

De MFM ondersteunt configuratie via LoRaWAN downlink berichten. Alle commando's kunnen op elke FPort worden verzonden.

7.1 Beschikbare Downlink Commando's

Commando	Bytes	Beschrijving
Interval wijzigen	`10 XX XX`	Meetinterval in seconden (big-endian)
Module commando	`11 AA CC [DD...]`	Commando naar sensormodule
Tandental wijzigen	`12 XX`	Aantal tanden meetwiel
Herstart forceren	`DE AD`	Reset na 5 seconden

7.2 Meetinterval Aanpassen

Formaat: `0x10 <high byte> <low byte>`

Voorbeeld: Interval naar 10 minuten (600 seconden)

```
600 = 0x0258
Downlink bytes: 10 02 58
```

Grenzen:

Parameter	Waarde
Minimum	20 seconden
Maximum	4270 seconden (~71 minuten)

Parameter	Waarde
Standaard	300 seconden (5 minuten)

TTN Console:

1. Ga naar je device
2. Klik "Messaging" -> "Downlink"
3. Payload type: "Bytes"
4. Vul in: `10 02 58`
5. Klik "Schedule downlink"

7.3 Tandental Aanpassen

Formaat: `0x12 <aantal>`

Dit configureert het aantal tanden op het meetwiel voor correcte RPM berekening.

Voorbeeld: 60 tanden

```
Downlink bytes: 12 3C
```

Standaardwaarde: 91 tanden

7.4 Sensormodule Commando

Formaat: `0x11 <module adres> <commando> [parameters...]`

Stuurt een SMBus commando door naar de aangesloten sensormodule.

Voorbeeld:

```
Module adres: 0x36
Commando: 0x20
Parameter: 0x01

Downlink bytes: 11 36 20 01
```

7.5 Device Herstarten

Formaat: `0xDE 0xAD`

Forceert een volledige reset van het device. De MFM zal:

1. 5 seconden wachten
2. Resetten
3. Opnieuw joinen met het LoRaWAN netwerk

Gebruik: Herstel na configuratiewijzigingen of problemen.

8. Debugging en Troubleshooting

8.1 Seriële Debug Output

De MFM stuurt uitgebreide debug informatie via UART wanneer `DEBUG` is gedefinieerd.

Verbinding maken:

1. Sluit FTDI adapter aan op de debug header
2. Open een seriële terminal op 115200 baud

```
# PlatformIO serial monitor
pio device monitor

# Of met screen (Linux/Mac)
screen /dev/ttyUSB0 115200
```

Voorbeeld debug output:

```
Build at: Feb  4 2026 14:17:32
[0] EV_JOINING
[5234] EV_JOINED
[5235] job_pingVersion
[50892] job_performMeasurements
[60893] job_fetchAndSend
[60893] I2C read result: 0 count: 8
[60894] Data: 00 04 93 E0 00 00 01 2C
[60895] Window milliseconds: 300000  Pulse count: 300
[60896] LoRaWAN payload FPort 3: 0A
[60897] Measurement scheduled: 4915200
```

8.2 Foutcodes

Debug Message	Betekenis	Oplossing
`Invalid EEPROM, did you flash the EEPROM?`	MAGIC bytes niet gevonden	Programmeer EEPROM met credentials
`EV_JOIN_TXCOMPLETE`	Join mislukt (geen ACK)	Controleer credentials en gateway bereik
`TXRX Pending...`	Vorige TX nog bezig	Verhoog meetinterval
`I2C read result: 1`	Sensor niet bereikbaar	Controleer I2C verbinding

8.3 LED Indicaties

LED Patroon	Status

LED Patroon	Status
Kort flikkeren bij opstarten	Normaal
Continu aan na EEPROM write	EEPROM programmering succesvol

8.4 Veelvoorkomende Problemen

Device joint niet:

1. Controleer of credentials correct zijn ingevoerd (byte order!)
2. Controleer gateway bereik
3. Verifieer dat device is geregistreerd in TTN
4. Wacht minimaal 5 minuten (join backoff)

Geen sensor data:

1. Controleer I2C verbindingen (SDA/SCL)
2. Verifieer sensormodule voeding
3. Check sensormodule adres (standaard 0x36)

Verkeerde RPM waarden:

1. Controleer tandental configuratie
2. Verifieer pulstellermodule werking
3. Check meetinterval (te kort = geen pulsen)

---

9. Technische Referentie

9.1 Pinout MFM (ATmega1284P)

Functie	Pin	Arduino Pin
Peripheral Power	PD4	20
JSN TX	PD2	10
JSN RX	PD3	11
OneWire	PD4	12
Buzzer	PA3	17
NSS (LoRa)	PB0	24
RST (LoRa)	PB1	25
DIO0	PD2	2
DIO1	PD3	3
DIO2	PD4	4

9.2 I2C/SMBus Communicatie

Parameter	Waarde
Bus snelheid	80 kHz
Sensormodule adres	0x36
Protocol	SMBus Block Read/Write

9.3 LoRaWAN Parameters

Parameter	Waarde
Activation	OTAA
Region	EU868
Spreading Factor	SF7-SF12 (ADR)
TX Power	14 dBm
Link Check	Disabled
ADR	Enabled

9.4 Uplink Payload Formaten

FPort 2 - Version Ping:

```
Byte 0:    0x10 (versie indicator)
Byte 1-2:  Firmware versie (big-endian)
Byte 3-4:  Hardware versie (big-endian)
```

FPort 3 - RPM Meting:

```
Byte 0:    RPM waarde (0-255)
```

9.5 Versie Encoding

Versienummers worden gecodeerd in 16 bits:

```
Bit 15:    Proto (0=dev, 1=release)
Bit 14-10: Major versie (0-31)
Bit 9-5:   Minor versie (0-31)
Bit 4-0:   Patch versie (0-31)
```

Voorbeeld: Versie 3.2.1 release = `0xC041`

Appendix A: Quick Reference Card

```
╔═══════════════════════════════════════════════════════════╗
║                 MultiFlexMeter Quick Reference            ║
╠═══════════════════════════════════════════════════════════╣
║ BUILD & UPLOAD                                            ║
║   pio run                    Build firmware               ║
║   pio run -t upload          Build and upload             ║
║   git submodule update --init  Initialize LMIC library    ║
╠═══════════════════════════════════════════════════════════╣
║ DOWNLINK COMMANDS                                         ║
║   10 XX XX     Set interval (seconds, big-endian)         ║
║   11 AA CC DD  Module command (addr, cmd, params)         ║
║   12 XX        Set wheel teeth count                      ║
║   DE AD        Force rejoin                               ║
╠═══════════════════════════════════════════════════════════╣
║ SERIAL DEBUG                                              ║
║   Baudrate: 115200                                        ║
║   pio device monitor                                      ║
╠═══════════════════════════════════════════════════════════╣
║ EEPROM BYTE ORDER                                         ║
║   APP_EUI:  LSB first (reverse from TTN)                  ║
║   DEV_EUI:  LSB first (reverse from TTN)                  ║
║   APP_KEY:  MSB first (same as TTN)                       ║
╚═══════════════════════════════════════════════════════════╝
```

Dit document is onderdeel van het Malen de Molens project - HBO ICT Bedrijfsproject Semester 7