```
#include <CommunicatieLib.h>
#include <MD_MAX72xx.h>
#include <RFID_Library.h>
#include "RFIDWriter.h"
#include <SPI.h>
#include <MFRC522.h>



/* ================== LED MATRIX ================== */
#define HARDWARE_TYPE MD_MAX72XX::FC16_HW
#define DATA_PIN 12
#define CLK_PIN  13
#define CS_PIN   14
#define MAX_DEVICES 1

MD_MAX72XX mx(HARDWARE_TYPE, DATA_PIN, CLK_PIN, CS_PIN, MAX_DEVICES);

/* ================== COMMUNICATIE ================== */
CommunicatieLib communicatie(1, 2); // Tx, Rx
Message msg;

/* ================== GAME VARIABELEN ================== */
int a = 0;
int b = 0;
bool aGelezen = false;
bool bGelezen = false;

bool operatorBekend = false;
```

```
Operators ontvangenOperator;


/* ================== LED PATRONEN ================== */

uint8_t plusP[8] = {

 0b00011000,0b00011000,0b00011000,0b11111111,0b11111111,
0b00011000,0b00011000,0b00011000

};

uint8_t minP[8] = {

  0b00000000,0b00000000,0b00000000,0b11111111,0b11111111,0b00000000,0b0000
0000, 0b00000000

};

uint8_t keerP[8] = {

  0b10000001,0b01000010,0b00100100,0b00011000,0b00011000,0b00100100,0b0100
0010,0b10000001

};

uint8_t gedeeldP[8] = {

  0b00000000,0b00011000,0b00011000,0b00000000,0b00000000,0b00011000,0b0001
1000,0b00000000

};


uint8_t *activePattern = nullptr;

unsigned long showUntil = 0;


/* ================== FUNCTIES ================== */


int rfidNaarGetal(byte *uid, byte uidSize) {

  int waarde = 0;

  for (byte i = 0; i < uidSize; i++) {

    waarde += uid[i];
```

```
  }
  return waarde % 10;
}

int berekenSom(int x, int y, Operators op) {
  switch (op) {
    case OP_PLUS:    return x + y;
    case OP_MIN:     return x - y;
    case OP_KEER:    return x * y;
    case OP_GEDEELD: return (y != 0) ? x / y : 0;
  }
  return 0;
}

void toonOperator(Operators op, unsigned long duurMs) {
  switch (op) {
    case OP_PLUS:    activePattern = plusP; break;
    case OP_MIN:     activePattern = minP; break;
    case OP_KEER:    activePattern = keerP; break;
    case OP_GEDEELD: activePattern = gedeeldP; break;
  }
  showUntil = millis() + duurMs;
}

void showPattern(uint8_t pattern[]) {
  for (int i = 0; i < 8; i++) {
    mx.setRow(0, i, pattern[i]);
  }
```

```
}

/* ================== SETUP ================== */
void setup() {
  mx.begin();
  mx.clear();
  communicatie.sendPing();
}

/* ================== LOOP ================== */
void loop() {

  /* ---- Communicatie ---- */
  if (communicatie.receive(msg, 2000)) {
    switch (msg.type) {

      case MSG_PING:
        communicatie.sendAck(msg.msgId);
        break;

      case MSG_GAME_DATA:
      ontvangenOperator =
        Operators op1 = static_cast<Operators>((msg.data[0] << 8) | msg.data[1]);
        Operators op2 = static_cast<Operators>((msg.data[2] << 8) | msg.data[3]);
        Operators op3 = static_cast<Operators>((msg.data[4] << 8) | msg.data[5]);
        OperatorBekend = true;
         break;
      case MSG_GAME_WIN:
```

```
    uint8_t playerNumber = msg.data[0]; // doe er wat leuks mee
    break;
  }
}

/* ---- RFID ---- */
if (rfidKaartGelezen) {
  int getal = rfidNaarGetal(uid, uidSize);

  if (!aGelezen) {
    a = getal;
    aGelezen = true;
  } else if (!bGelezen) {
    b = getal;
    bGelezen = true;
  }
}

/* ---- Berekening ---- */
if (aGelezen && bGelezen && operatorBekend) {
  toonOperator(ontvangenOperator, 2000);
  delay(2000);

  int resultaat = berekenSom(a, b, ontvangenOperator);
  // hier kun je resultaat vergelijken / verzenden

  aGelezen = false;
  bGelezen = false;
```

```
    operatorBekend = false;
  }

  /* ---- LED refresh ---- */
  if (activePattern) {
    showPattern(activePattern);
    if (millis() > showUntil) {
      mx.clear();
      activePattern = nullptr;
    }
  }
}
```