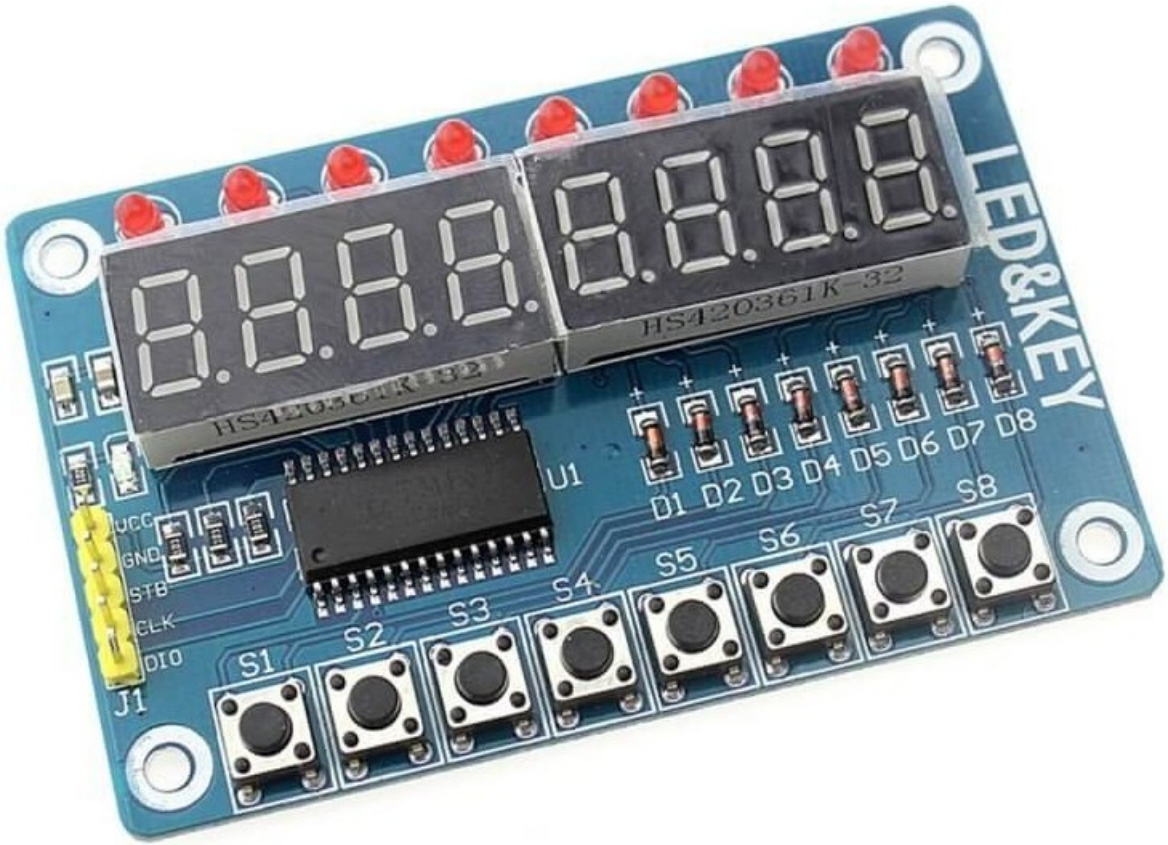
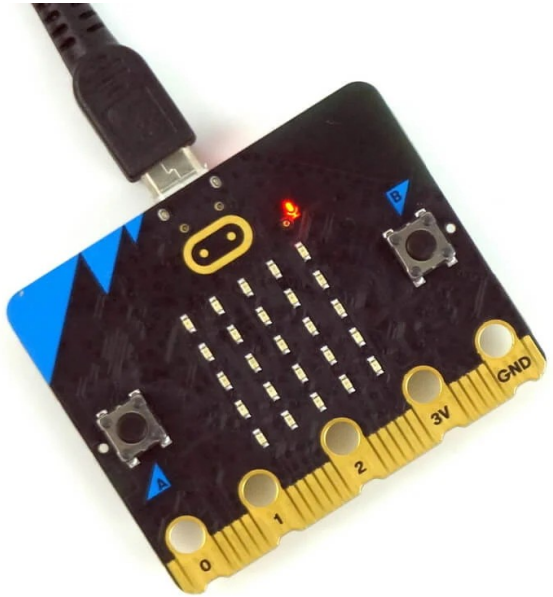
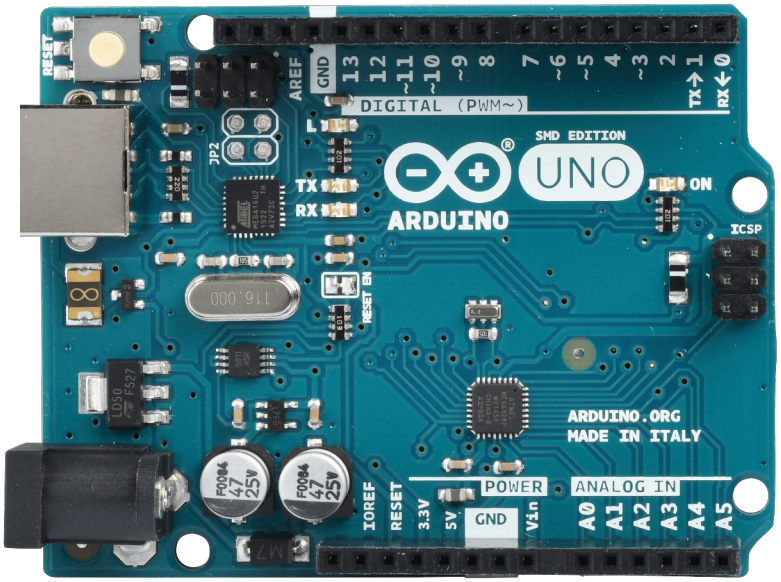
A TM1638 LED&KEY kártya használata





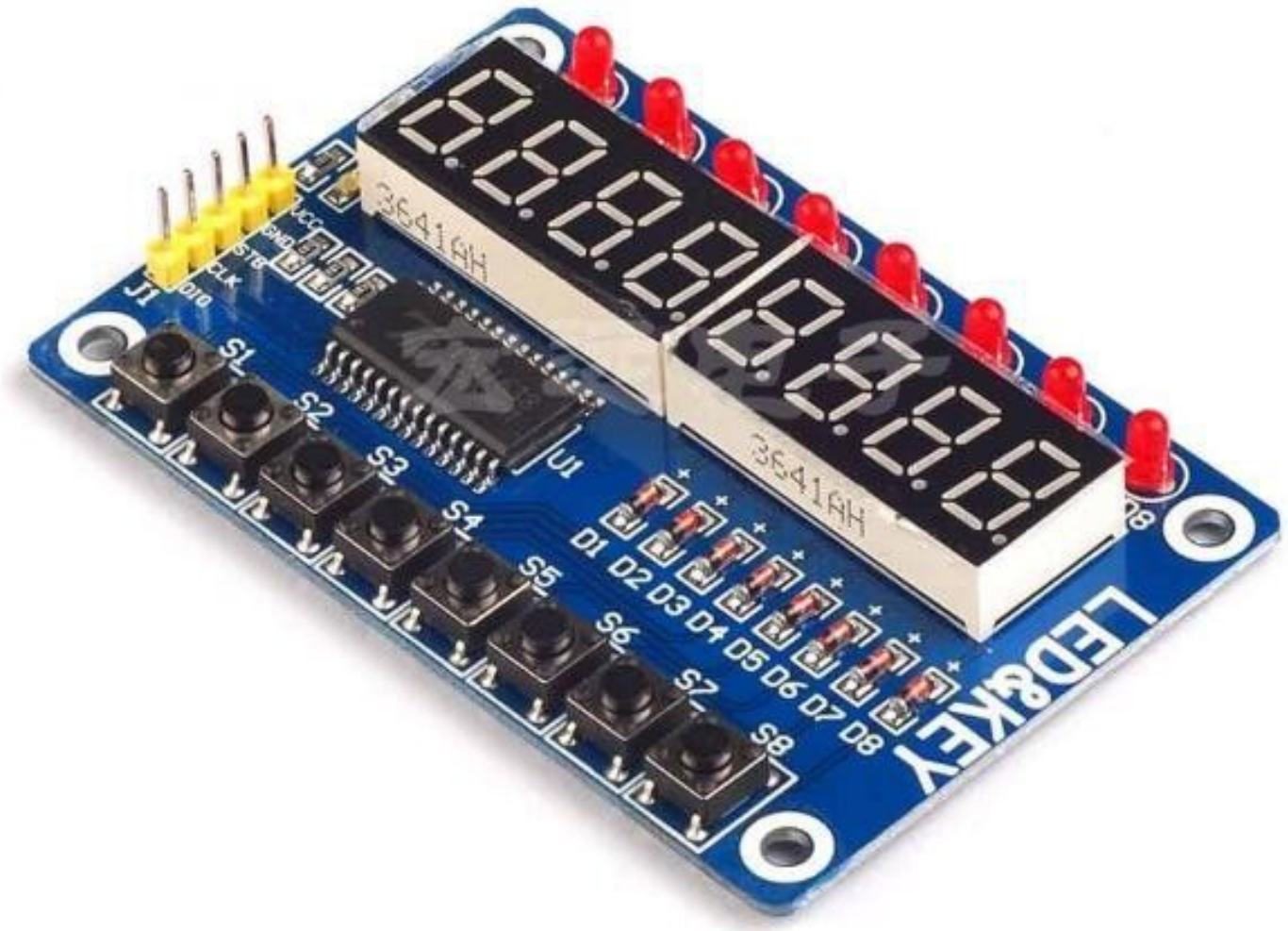


Felhasznált és ajánlott irodalom

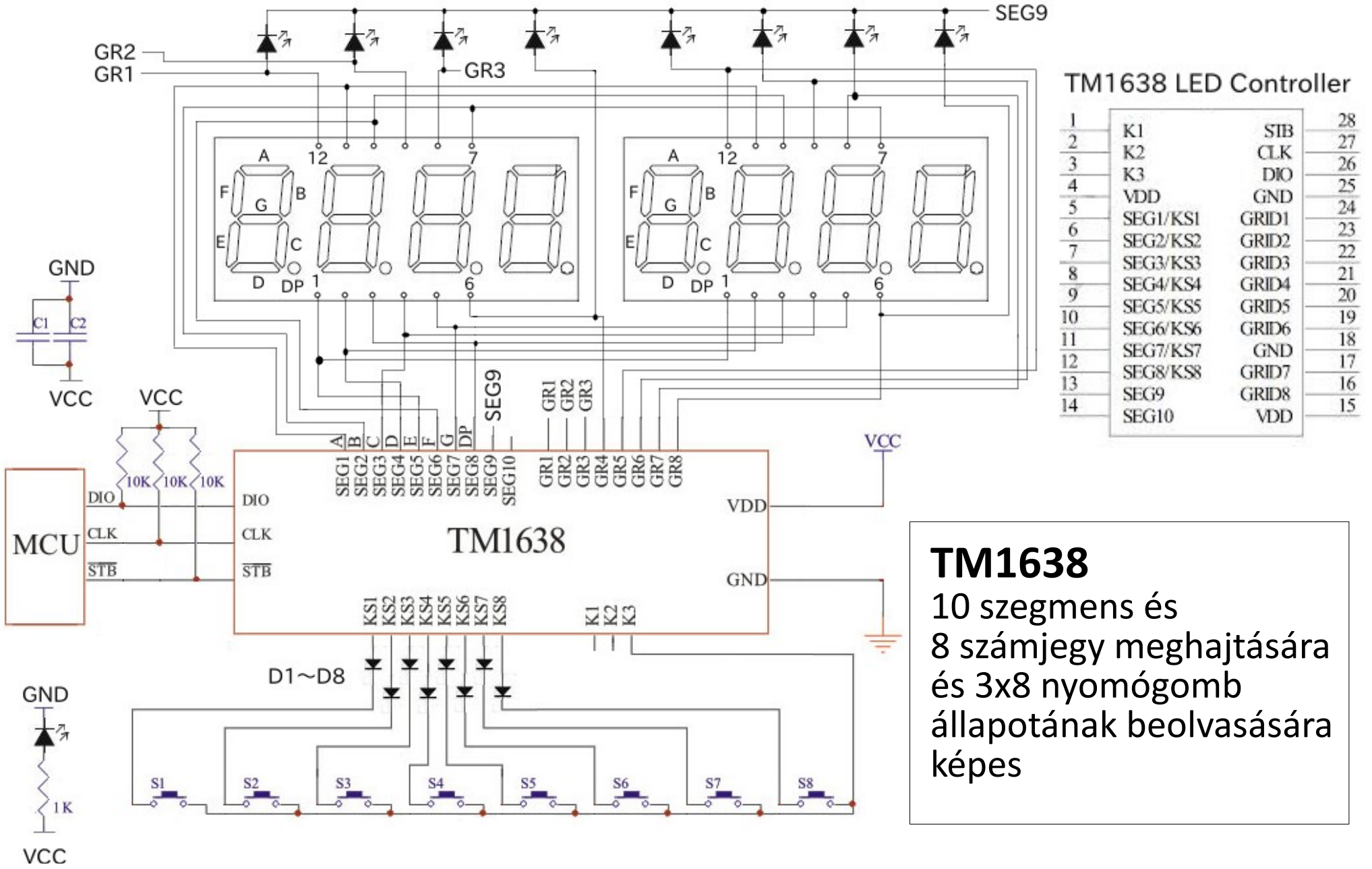
- Micro:bit botorkálás: Segédanyagok
- Dr. Abonyi-Tóth Andor: Programozzunk micro:biteket!
- Cytron Technologies: EDU:BIT könyv
- BBC Micro:bit: Microbit.org
- Micro:bit developer community: Circuit Schematics
- Microsoft MakeCode: Documentation
- BBC Micro:bit Python Reference
- BBC Micro:bit MicroPython documentation
- Codewith.mu: Mu and micro:bit
- Martyn Davies: TM1638 Display Module with BBC micro:bit
- Makerguides: How to use Arduino UNO with TM1638 7-Segment LED Driver

Nyolcszámjegyű kijelző TM1638 vezérlővel

- A LED & KEY kártya a **TM1638** vezérlőt használja, amely a 8-számjegyű 7-szegmenses kijelző mellett 8 db LED-et és 8 db nyomógombot is kezel



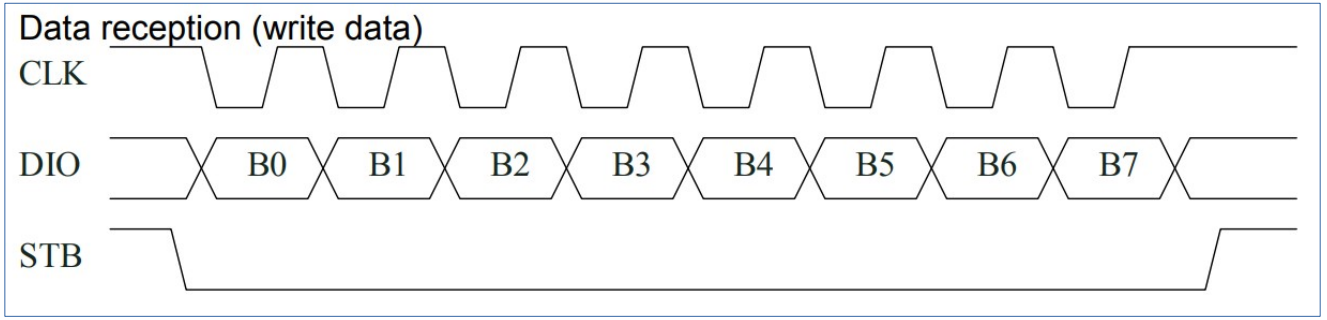
Kapcsolási vázlat



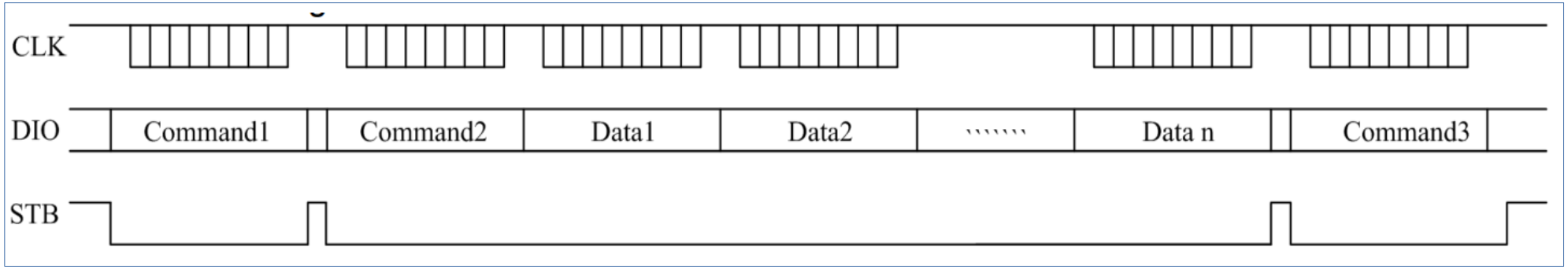
TM1638
10 szegmens és
8 számjegy meghajtására
és 3x8 nyomógomb
állapotának beolvasására
képes

Kommunikáció és parancsok

- A kommunikáció az **SPI** Mode 3-hoz hasonlóan zajlik, de LSBFIRST módon (a legkisebb helyiértékű bit megy ki elsőként)



- A kommunikáció háromféle parancsot használ:
  1. adatmód beállítás, 2. cím beállítás és 3. kijelzési mód vezérlés



- A strobe vonal (STB) lehúzása után mindig parancsot vár a vezérlő
- A címbeállítás parancsot legfeljebb 16 adatbájt követheti

TM1638 adatmód beállító parancsok

- Tipikus adatmód beállítások:
  - **0x40** adatküldés automatikus címnöveléssel
  - **0x44** adatküldés fix címre
  - **0x42** nyomógomb állapotok lekérdezése

B7	B6	Instruction
0	1	Setting of Data Command
1	0	Setting of Display Control Command
1	1	Setting of Address Command

MSB LSB

B7	B6	B5	B4	B3	B2	B1	B0	Function	Description
0	1	Unrelated item, fill 0				0	0	Setting of data read-write mode	Write data to the display register
0	1					1	0		Read key scanning data
0	1				0			Sett address increment mode	Auto increment
0	1				1				Fixed address
0	1			0				Test mode setting (for internal use)	Normal mode
0	1			1					Test mode

Címbeállító parancsok

- A címbeállító parancsok **0xC0** és **0xCF** közötti értékek lehetnek
- A 10 szegmensvezérlő jel két-két bájtot foglal el (ebből a páros című bájtok a számkijelző szegmenseket, a páratlan számon levő bájtok legalsó bitje pedig a különálló LED-eket vezérlik
- 0xC0 a bal szélső számjegyet, 0xC1pedig a bal szélső LED-et vezérli

MSB B7	B6	B5	B4	B3	B2	B1	LSB B0	Display address
1	1	Unrelated item, fill 0		0	0	0	0	00H
1	1			0	0	0	1	01H
1	1			0	0	1	0	02H
1	1			0	0	1	1	03H
1	1			0	1	0	0	04H
1	1			0	1	0	1	05H
1	1			0	1	1	0	06H
1	1			0	1	1	1	07H
1	1			1	0	0	0	08H
1	1			1	0	0	1	09H
1	1			1	0	1	0	0AH
1	1			1	0	1	1	0BH
1	1			1	1	0	0	0CH
1	1			1	1	0	1	0DH
1	1			1	1	1	0	0EH
1	1			1	1	1	1	0FH

SEG1	SEG2	SEG3	SEG4	SEG5	SEG6	SEG7	SEG8	SEG9	SEG10	X	X	X	X	X	X	
xxHL (low four)				xxHU (high four)				xxHL (low four)				xxHU (high four)				
B0	B1	B2	B3	B4	B5	B6	B7	B0	B1	B2	B3	B4	B5	B6	B7	
00HL				00HU				01HL				01HU				GRID1
02HL				02HU				03HL				03HU				GRID2
04HL				04HU				05HL				05HU				GRID3
06HL				06HU				07HL				07HU				GRID4
08HL				08HU				09HL				09HU				GRID5
0AHL				0AHU				0BHL				0BHU				GRID6
0CHL				0CHU				0DHL				0DHU				GRID7
0EHL				0EHU				0FHL				0FHU				GRID8

Kijelzési mód vezérló parancsok

- Ha a **B3** bit 0, a kijelző lekapcsol (pl. **0x80** parancskódra)
- A **0x88 – 0x8F** parancsok engedélyezik a kijelzést és a **B0-B2** bitek beállítják a fényerőt

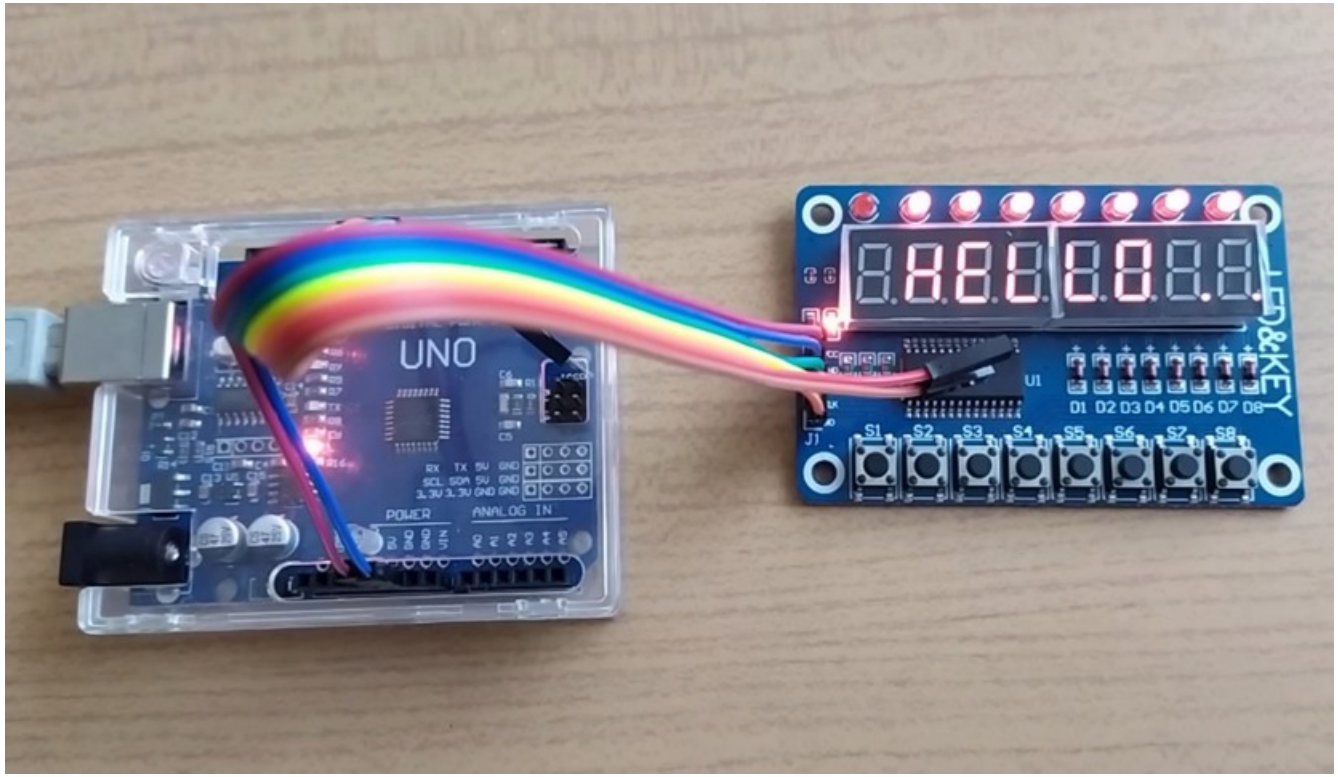
MSB LSB

B7	B6	B5	B4	B3	B2	B1	B0	Function	Description
1	0	Unrelated item, fill 0			0	0	0	Set the number of extinction	Set the pulse width to 1/16
1	0				0	0	1		Set the pulse width to 2/16
1	0				0	1	0		Set the pulse width to 4/16
1	0				0	1	1		Set the pulse width to 10/16
1	0				1	0	0		Set the pulse width to 11/16
1	0				1	0	1		Set the pulse width to 12/16
1	0				1	1	0		Set the pulse width to 13/16
1	0				1	1	1		Set the pulse width to 14/16
1	0			0				Setting of display switch	Display Off
1	0			1					Display ON

Arduino programok

- A **LED&KEY** kártyát gyakran használják Arduino perifériaként
- Használhatjuk a beépített **shiftOut()/shiftIn()** függvényeket is, de számos programkönyvtár is található ehhez a kártyához
- **Bekötés:** a szoftveres kezelés miatt bármelyik digitális kivezetést használhatjuk. A mellékelt példaprogramok az alábbi bekötést feltételezik

TM1638	Arduino
VCC	5V
GND	GND
STB	D8
CLK	D10
DIO	D9



Barebone programozás

- A How to use Arduino UNO with TM1638 7-Segment LED Driver című cikk egy olyan tesztprogramot mutat be, amely nem használ semmilyen programkönyvtárat a **TM1638** meghajtásához. Talán ez a legalkalmasabb mód, hogy a vezérlés részleteibe belelássunk
- **A program az alábbi lépésekből áll:**
  - **TM1638** inicializálás (regiszterek nullázása, max. fényerő beállítása)
  - Számláló mód: számjegyeket ír ki 0-tól 9-ig
  - Szöveg (HELLO) és LED futófény görgetése
  - **PRESS** és **buttonS** felirat váltogatása, s a lenyomott nyomógomb(ok)hoz rendelt LED(ek) kivilágítása, amíg a gomb(ok) lenyomott állapotban van(nak)

TM1638_test.ino – 6/1.

Forrás: How to use Arduino UNO with TM1638 7-Segment LED Driver

```
#define COUNTING_MODE 0
#define SCROLL_MODE 1
#define BUTTON_MODE 2

const int strobe = 8;     // STB to D8
const int clock = 10;     // CLK to D10
const int data = 9;       // DIO to D9

void sendCommand(uint8_t value) {           // for Command1 and Command3
  digitalWrite(strobe, LOW);
  shiftOut(data, clock, LSBFIRST, value);
  digitalWrite(strobe, HIGH);
}

void reset() {
  sendCommand(0x40);     // Set auto increment mode
  digitalWrite(strobe, LOW);
  shiftOut(data, clock, LSBFIRST, 0xc0);   // Set starting address to 0
  for (uint8_t i = 0; i < 16; i++) {
    shiftOut(data, clock, LSBFIRST, 0x00); // Clear data registers
  }
  digitalWrite(strobe, HIGH);
}
```

TM1638_test.ino – 6/2.

```
void setup() {
  pinMode(strobe, OUTPUT);
  pinMode(clock, OUTPUT);
  pinMode(data, OUTPUT);
  sendCommand(0x8f);          // Set maximum display brightness
  reset();
}

void loop() {
  static uint8_t mode = COUNTING_MODE;
  switch (mode) {
    case COUNTING_MODE:
      mode += counting();
      delay(500);
      break;
    case SCROLL_MODE:
      mode += scroll();
      break;
    case BUTTON_MODE:
      buttons();
      break;
  }
  delay(200);
}
```

TM1638_test.ino – 6/3.

- Nullától 9-ig számolva a soron következő számjegyet megjelenítjük mind a 8 pozícióban (’0000 0000’ – ’9999 9999’)
- A seg9 és seg10 vezérlő biteket töröljük, tehát LED1 – LED8 kialszik (ha eredetileg égett volna)

```
bool counting() {
                /*0*/ /*1*/ /*2*/ /*3*/ /*4*/ /*5*/ /*6*/ /*7*/ /*8*/ /*9*/
  uint8_t digits[] = { 0x3f,0x06,0x5b,0x4f,0x66,0x6d,0x7d,0x07,0x7f,0x6f };
  static uint8_t digit = 0;
  sendCommand(0x40);
  digitalWrite(strobe, LOW);
  shiftOut(data, clock, LSBFIRST, 0xc0); // Set starting address to 0
  for (uint8_t position = 0; position < 8; position++)   {
    shiftOut(data, clock, LSBFIRST, digits[digit]);   // write seg1 - seg8
    shiftOut(data, clock, LSBFIRST, 0x00);            // clear seg9 & seg10
  }
  digitalWrite(strobe, HIGH);
  digit = ++digit % 10;
  return digit == 0;
}
```

TM1638_test.ino – 6/4.

```
bool scroll(){
  uint8_t scrollText[] =  {
    /* */ /* */ /* */ /* */ /* */ /* */ /* */ /* */
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
    /*H*/ /*E*/ /*L*/ /*L*/ /*O*/ /*.*/ /*.*/ /*.*/
    0x76, 0x79, 0x38, 0x38, 0x3f, 0x80, 0x80, 0x80,
    /* */ /* */ /* */ /* */ /* */ /* */ /* */ /* */
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
    /*H*/ /*E*/ /*L*/ /*L*/ /*O*/ /*.*/ /*.*/ /*.*/
    0x76, 0x79, 0x38, 0x38, 0x3f, 0x80, 0x80, 0x80,
  };
  static uint8_t index = 0;
  uint8_t scrollLength = sizeof(scrollText);
  sendCommand(0x40);
  digitalWrite(strobe, LOW);
  shiftOut(data, clock, LSBFIRST, 0xc0);

  for (int i = 0; i < 8; i++) {
    uint8_t c = scrollText[(index + i) % scrollLength];
    shiftOut(data, clock, LSBFIRST, c);
    shiftOut(data, clock, LSBFIRST, c != 0 ? 1 : 0);  // light up LED for nonzero char
  }

  digitalWrite(strobe, HIGH);
  index = ++index % (scrollLength << 1);
  return index == 0;
}
```

SCROLL

A ’ HELLO…’ szöveg görgetése

A nem üres karakterek fölött kigyújtjuk a LED-et

TM1638_test.ino – 6/5.

```
void buttons() {
  uint8_t promptText[] =   {
    /*P*/ /*r*/ /*E*/ /*S*/ /*S*/ /* */ /* */ /* */
    0x73, 0x50, 0x79, 0x6d, 0x6d, 0x00, 0x00, 0x00,
    /* */ /* */ /* */ /* */ /* */ /* */ /* */ /* */
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
    /*b*/ /*u*/ /*t*/ /*t*/ /*o*/ /*n*/ /*S*/ /* */
    0x7c, 0x1c, 0x78, 0x78, 0x5c, 0x54, 0x6d, 0x00,
    /* */ /* */ /* */ /* */ /* */ /* */ /* */ /* */
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  };
  static uint8_t block = 0;
  uint8_t textStartPos = (block / 4) << 3;
  for (uint8_t position = 0; position < 8; position++) {
    sendCommand(0x44);
    digitalWrite(strobe, LOW);
    shiftOut(data, clock, LSBFIRST, 0xC0 + (position << 1));
    shiftOut(data, clock, LSBFIRST, promptText[textStartPos + position]);
    digitalWrite(strobe, HIGH);
  }
  block = (block + 1) % 16;
  uint8_t buttons = readButtons();
  for (uint8_t position = 0; position < 8; position++) {
    uint8_t mask = 0x1 << position;
    setLed(buttons & mask ? 1 : 0, position);
  }
}
```

A **PRESS** és **buttonS** feliratok váltogatása, s a lenyomott nyomógombokhoz rendelt LED-ek kivilágítása, amíg a gombok lenyomott állapotban vannak

TM1638_test.ino – 6/6.

- Függvények a nyomógomb állapoto beolvasására, illetve egy LED állapotának beállítására

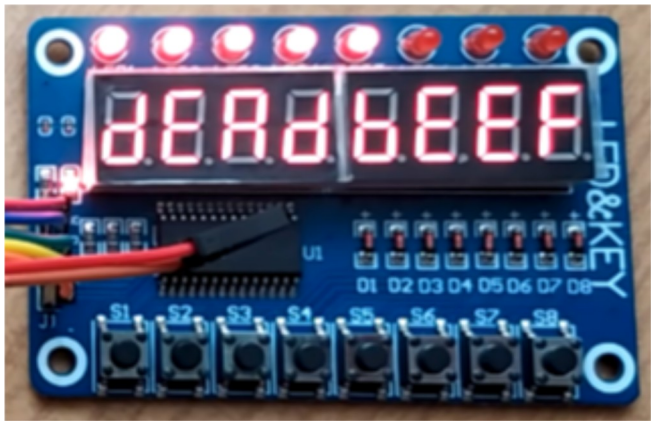
```
uint8_t readButtons(void) {
  uint8_t buttons = 0;
  digitalWrite(strobe, LOW);
  shiftOut(data, clock, LSBFIRST, 0x42);
  pinMode(data, INPUT);
  for (uint8_t i = 0; i < 4; i++)   {
    uint8_t v = shiftIn(data, clock, LSBFIRST) << i;
    buttons |= v;
  }
  pinMode(data, OUTPUT);
  digitalWrite(strobe, HIGH);
  return buttons;
}

void setLed(uint8_t value, uint8_t position) {   // value = 0..1, position = 0..7
  pinMode(data, OUTPUT);
  sendCommand(0x44);
  digitalWrite(strobe, LOW);
  shiftOut(data, clock, LSBFIRST, 0xC1 + (position << 1));
  shiftOut(data, clock, LSBFIRST, value);
  digitalWrite(strobe, HIGH);
}
```

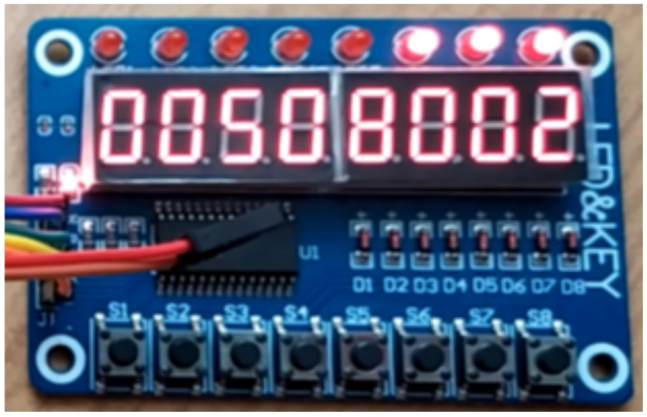
A TM1638plus programkönyvtár

- Gavin Lyons TM1638plus programkönyvtára Arduino környezethez készült, háromféle TM1638 alapú kártyát és hatféle mikrovezérlőt támogat (Arduino Uno & nano, Attiny85, ESP8266, ESP32, Teensy 4.0 és STM32F103C8 „blue pill”)
- Az API leírása itt található
- A legfontosabb tagfüggvények:
  - **displayBegin**() – inicializálás
  - **reset**() – a kijelző törlése
  - **brightness**(val) – a fényerő beállítása (val = 0..7)
  - **readButtons**() – a nyomógombok állapotának beolvasása
  - **setLED**(pos, val) – egy LED állapotának beállítása
  - **displayText**(*text) – szöveg kiíratása
  - **displayIntNum**(num) – szám kiíratása (bevezető nullákkal, vagy anélkül)
  - **displayHex**(pos, num) – szám kiíratása hexadecimálisan, adott pozícióba

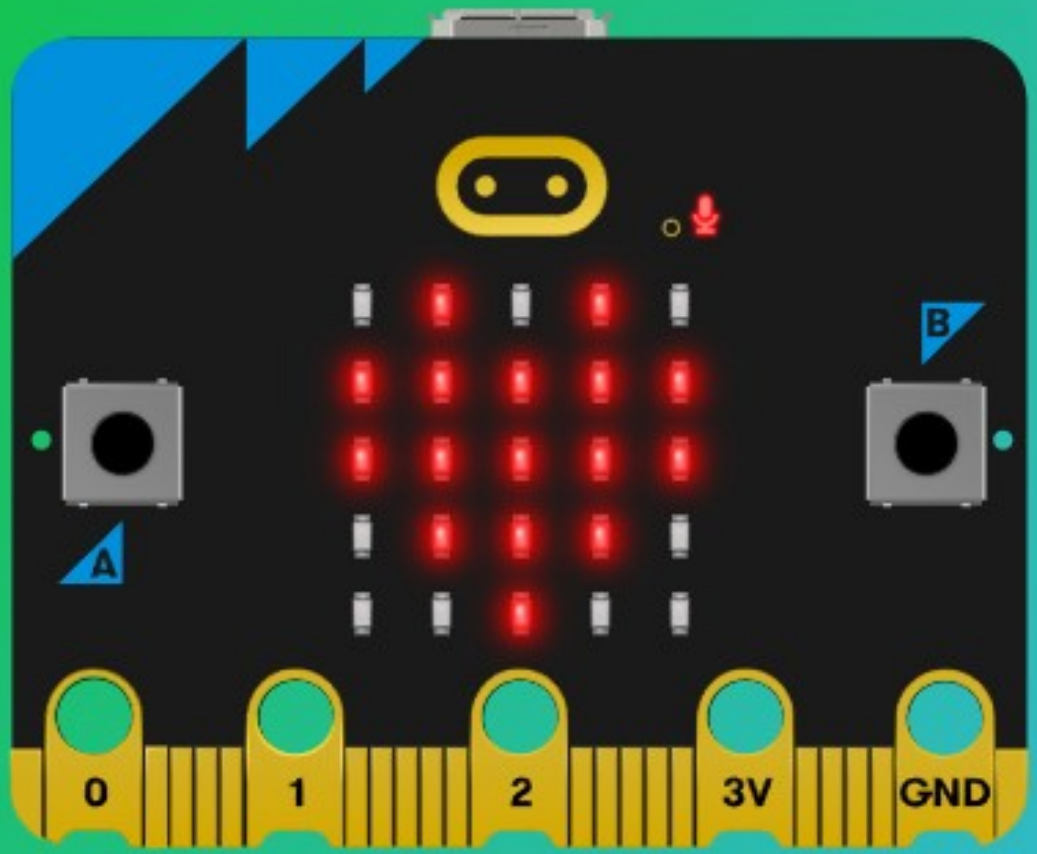
TM1638plus_demo.ino

```
#include <TM1638plus.h>
#define  STROBE_TM 8          // strobe = GPIO connected to strobe line of module
#define  CLOCK_TM 10          // clock = GPIO connected to clock line of module
#define  DIO_TM    9          // data = GPIO connected to data line of module
bool high_freq = false;       // for high freq CPU set to true

TM1638plus tm(STROBE_TM, CLOCK_TM , DIO_TM, high_freq);

void setup() {
  tm.displayBegin();
  tm.brightness(5);
}

void loop() {
  tm.displayText("DEADBEEF");
  for (int i=0; i<8; i++) {
    tm.setLED(i,1);           // Turn on LEDs
    delay(250);
  }
  delay(1000);
  tm.displayIntNum(millis());
  for (int i=0; i<8; i++) {
    tm.setLED(i,0);           // Turn off LEDs
    delay(250);
  }
  delay(1000);
}
```





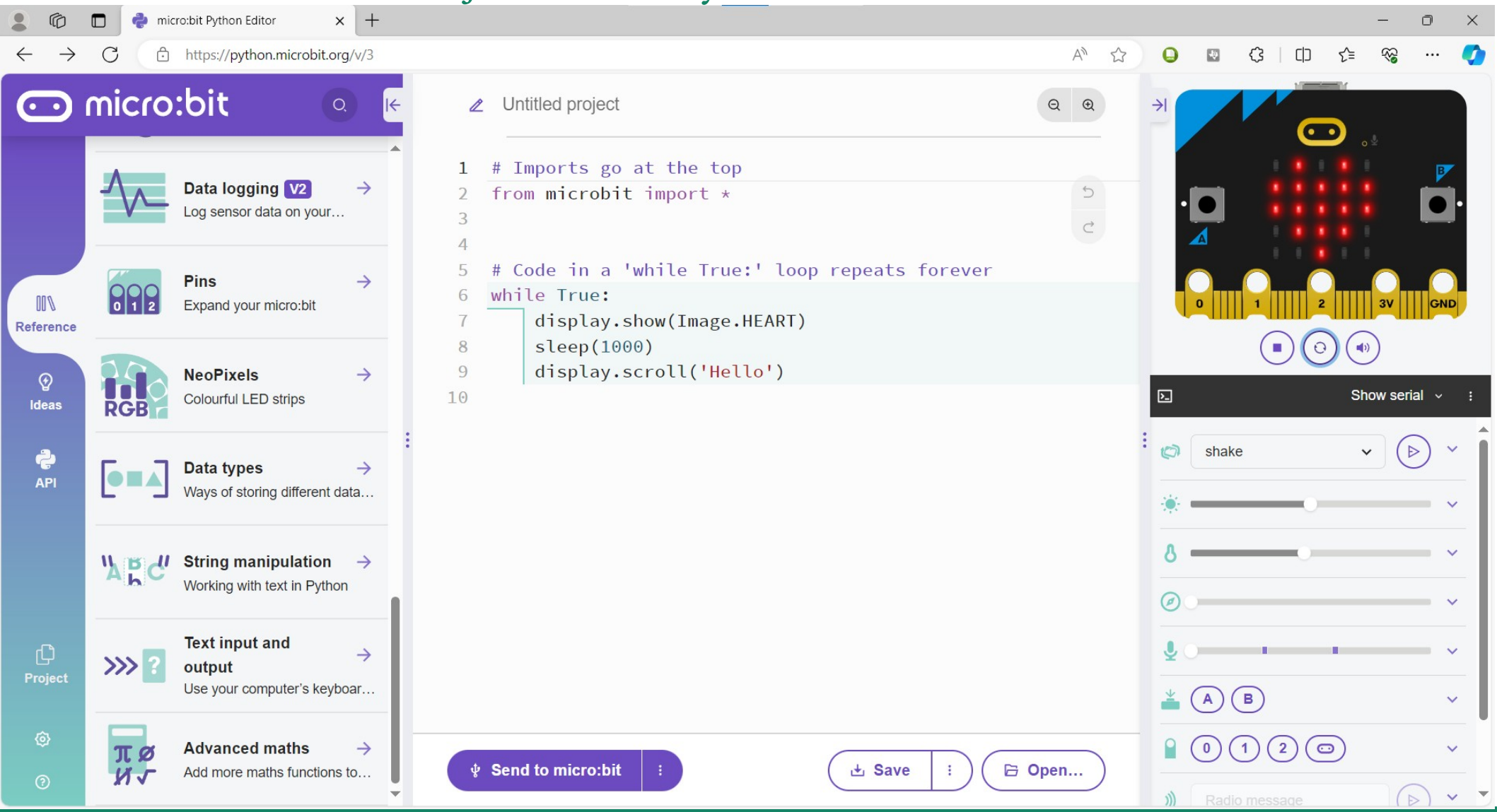
BBC micro:bit v2



Create | Learn | Code

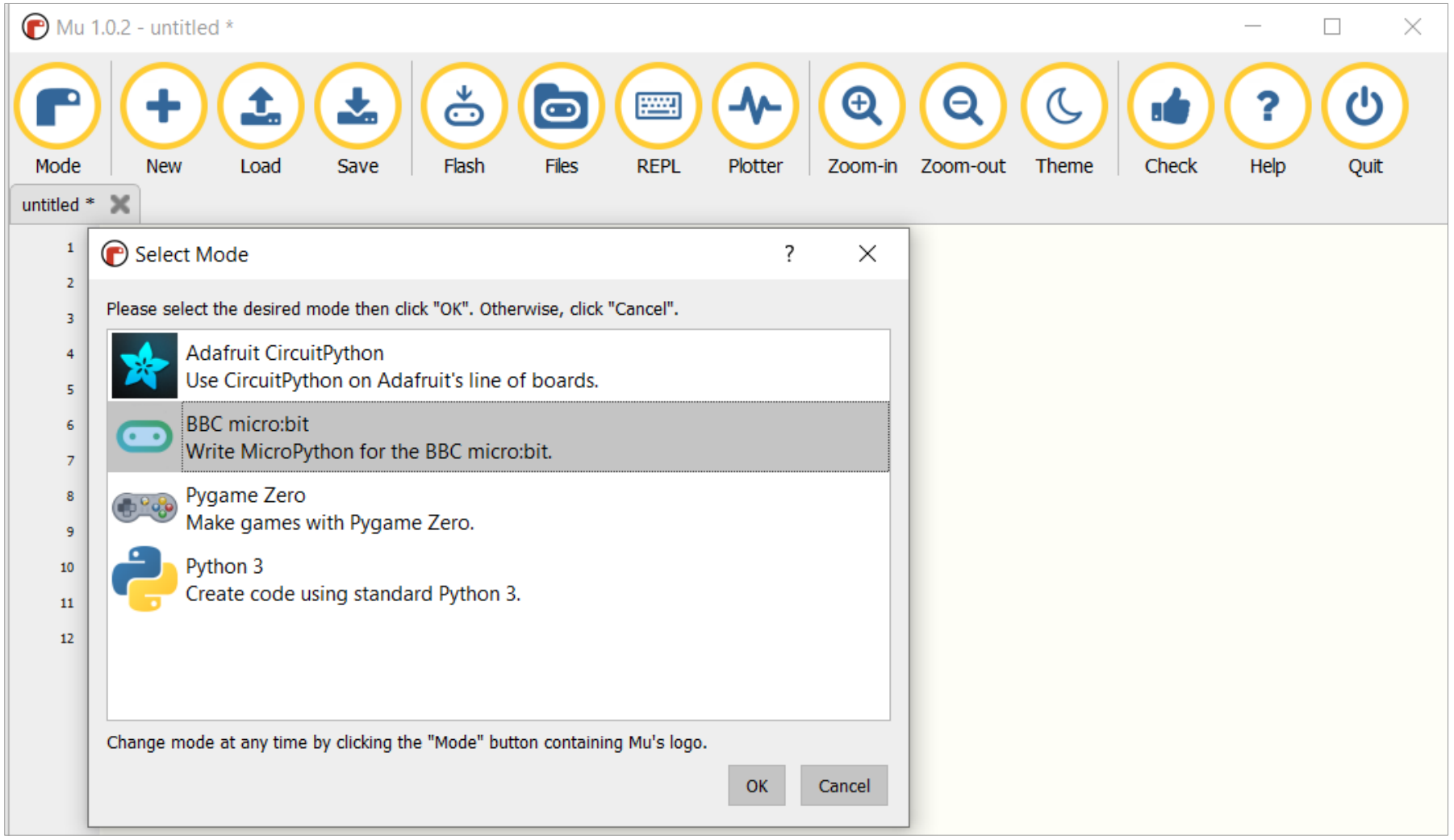
Emlékeztető: Python Editor

- A **micro:bit Python Editor** egy online, fejlett és könnyen használható kódfejlesztő környezet és szimulátor



Emlékeztető: Mu és a micro:bit kártya

- A Mu editor ***Mode*** gombjával ***BBC micro:bit*** üzemmódot is választhatunk Ezután a menüben megjelenő **Flash** gomb a programletöltés, a **REPL** az interaktív **MicroPython** futtatás, a **Files** pedig a filekezelő

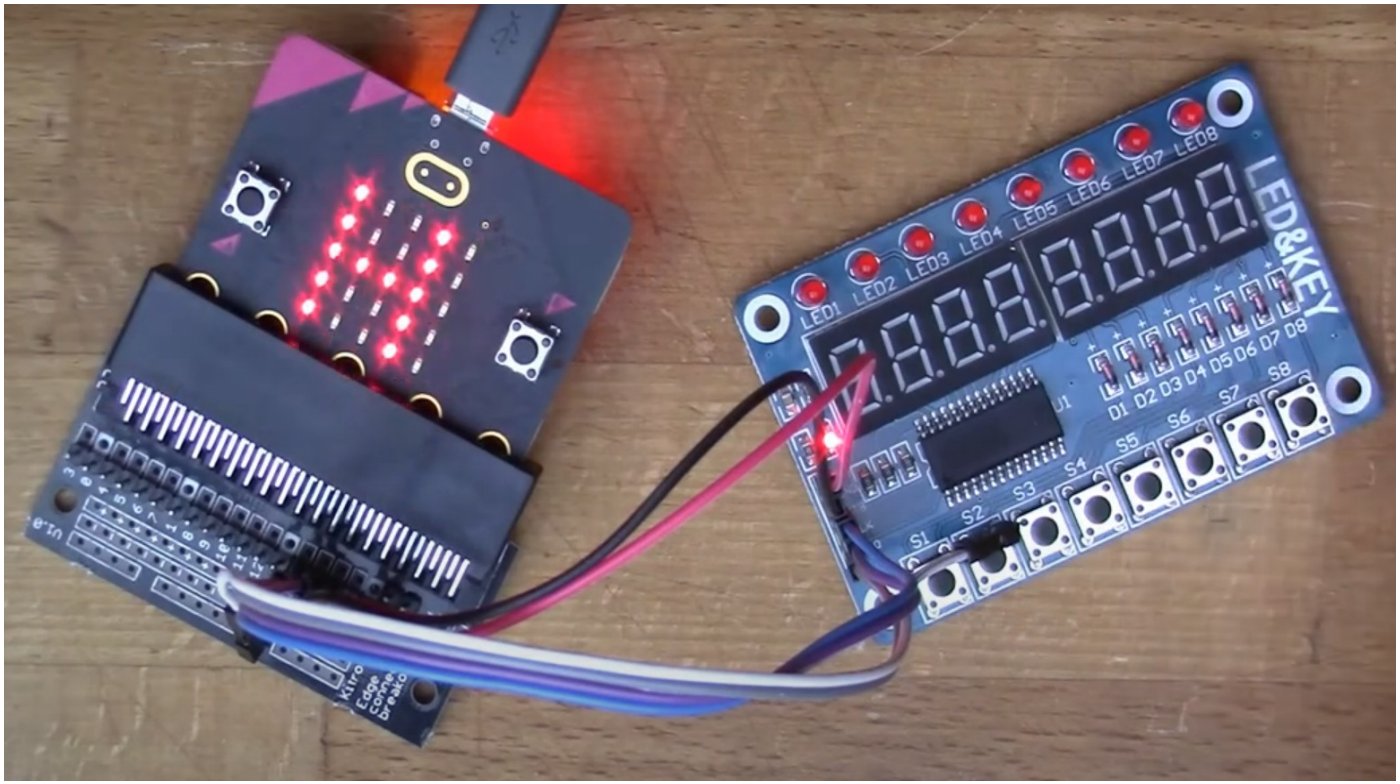


A TM1638 kártya Micro:bit v2 vezérlővel

- A **Micro:bit v2** kártya és a **TM1638 LED&KEY** kártya összekötéséhez kell még egy **Kitronik Edge Connector Breakout Board** (élcsatlakozóról tüskékre kivezető kártya) is

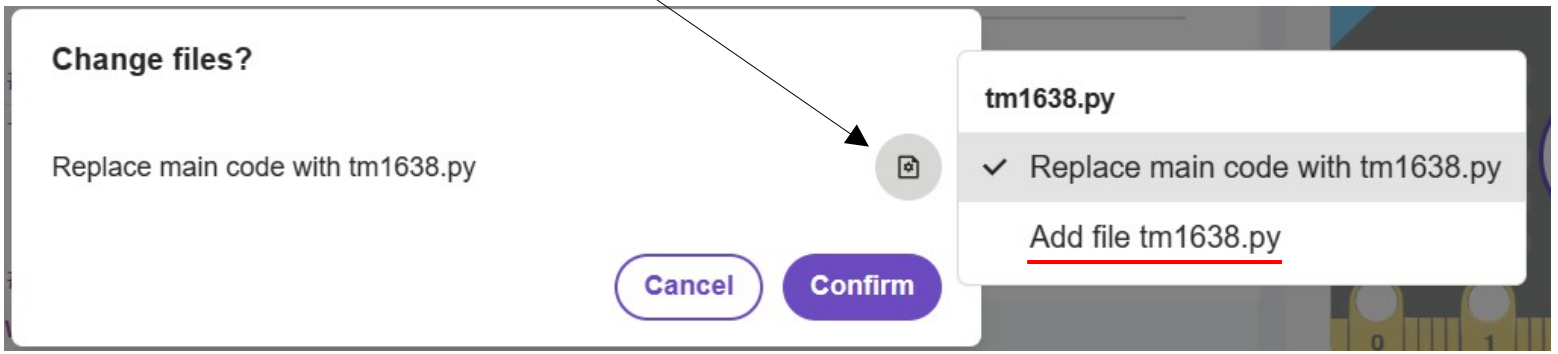
- Bekötési vázlat:

TM1638	Micro:bit
VCC	3V
GND	0V
STB	pin15
CLK	pin13
DIO	pin12



TM1638 vezérlés MicroPython programokkal

- Martyn Davies: TM1638 Display Module with BBC micro:bit bemutatja, hogy hogyan használhatjuk a **TM1638** vezérlővel megépített LED&KEY kártyát **Micro:bit** mikrovezérlővel
- Az általa adaptált MicroPython programkönyvtár itt található: **tm1638.py** (ez Mike Causer tm1638.py programkönyvtárának **Micro:bit**-re átdolgozott változata)
- Az online **Micro:bit Python Editor**-ban az Open File gombra kattintva tölthetjük fel a tm1638.py programkönyvtárat a projekthez
- A kis ikonra kattintva a felbukkanó mezőben válasszuk az *„Add file”* opciót!



tm1638_accelx_demo.py

- Az alábbi programocskában szövegkiírást, LED futófény animációt és előjeles számkiírást láthatunk
- A program a gyorsulásmérő x komponensét olvassa ki és jeleníti meg, így vízmértékként használhatjuk (0 körüli érték a vizszintes)

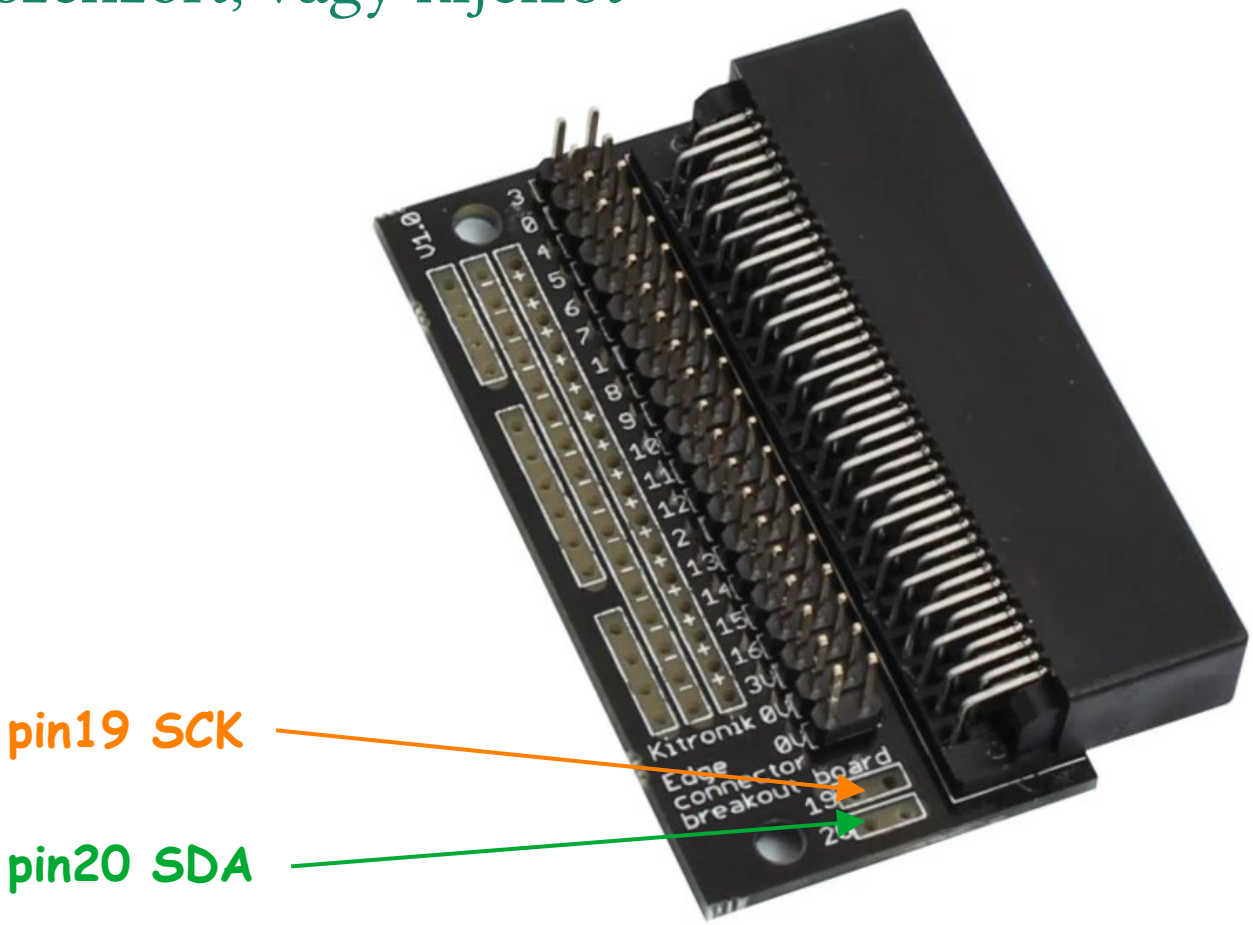
```
from microbit import *
from tm1638 import TM1638
tm = TM1638(stb=pin15, clk=pin14, dio=pin13)
tm.leds(0xFF)                                # Light up all LEDs
tm.show('Hello')                             # Display text
sleep(1000)
while True:
    tm.leds(0x92)                            # Animamate LED pattern
    sleep(200)
    tm.leds(0x49)
    sleep(200)
    tm.leds(0x24)
    sleep(200)
    x_strength = accelerometer.get_x() # Readd accelerator x data
    tm.number(x_strength)                    # Display as signed number
```

tm1638.py programkönyvtár

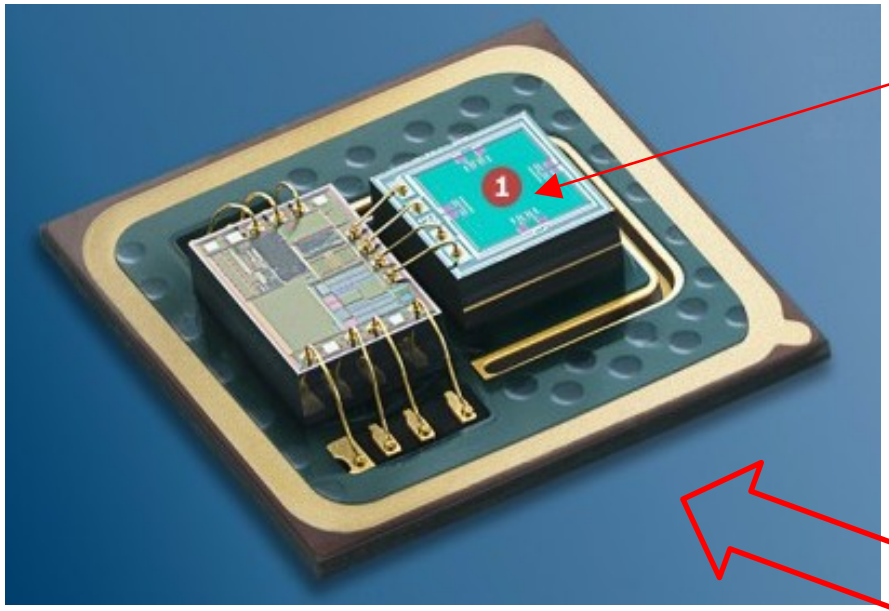
- A fontosabb metódusok:
  - **brightness**(val=None) – a fényerő beállítása (val = 0..7)
  - **clear**() - a kijelző törlése
  - **write**(data, pos=0) – adatregiszter beírás
  - **led**(pos, val) – egy LED beállítása (pos = 0..7, val = 0..1)
  - **leds**(val) – LED-ek beállítása egyszerre
  - **segments**(segments, pos=0) – szegmensek egyedi vezérlése
  - **keys**()- nyomóbomb állapotok beolvasása
  - **number**(num) – előjeles szám kiírása, jobbra igazítva
  - **hex**(val) – előjel nélküli szám hexadecimális kiíratása, jobbra igazítva
  - **show**(string, pos=0) – string kiíratása
  - **scroll**(string, delay=250) – szöveg görgetése jobbról balra
  - **temperature**(num, pos=0) – hőmérséklet kiíratása °C-vel együtt
  - **humidity**(num, pos=4) – relatív páratartalom kiírása rH jelzéssel

Az I2C busz kivezetései

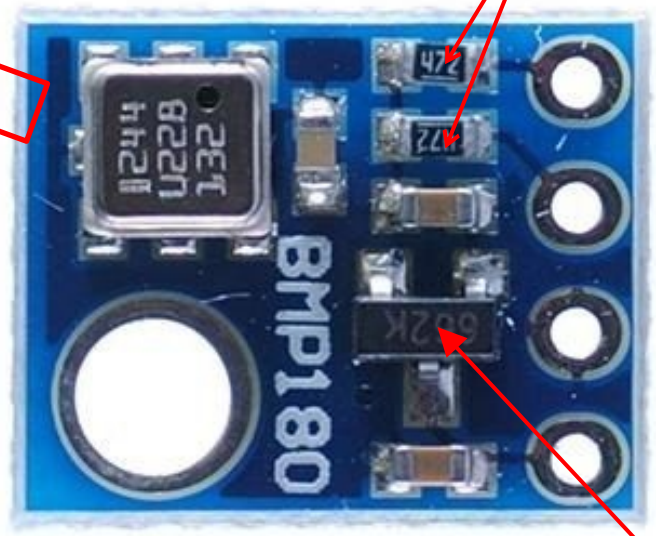
- A **Kitronik Edge Connector Breakout Board-on a** pin19 és pin20 tüskéi (az I2C busz SDA és SCK kivezetései) nincsenek beforrasztve, ezt nekünk kell megoldani, ha csatlakoztatni akarunk valamilyen I2C szenzort, vagy kijelzőt



Légnyomás mérése BMP180 szenzorral



Piezorezisztív nyúlásmérő, amely a nyomás hatására bekövetkező deformációt érzékeli



Bosch SensorTec BMP180

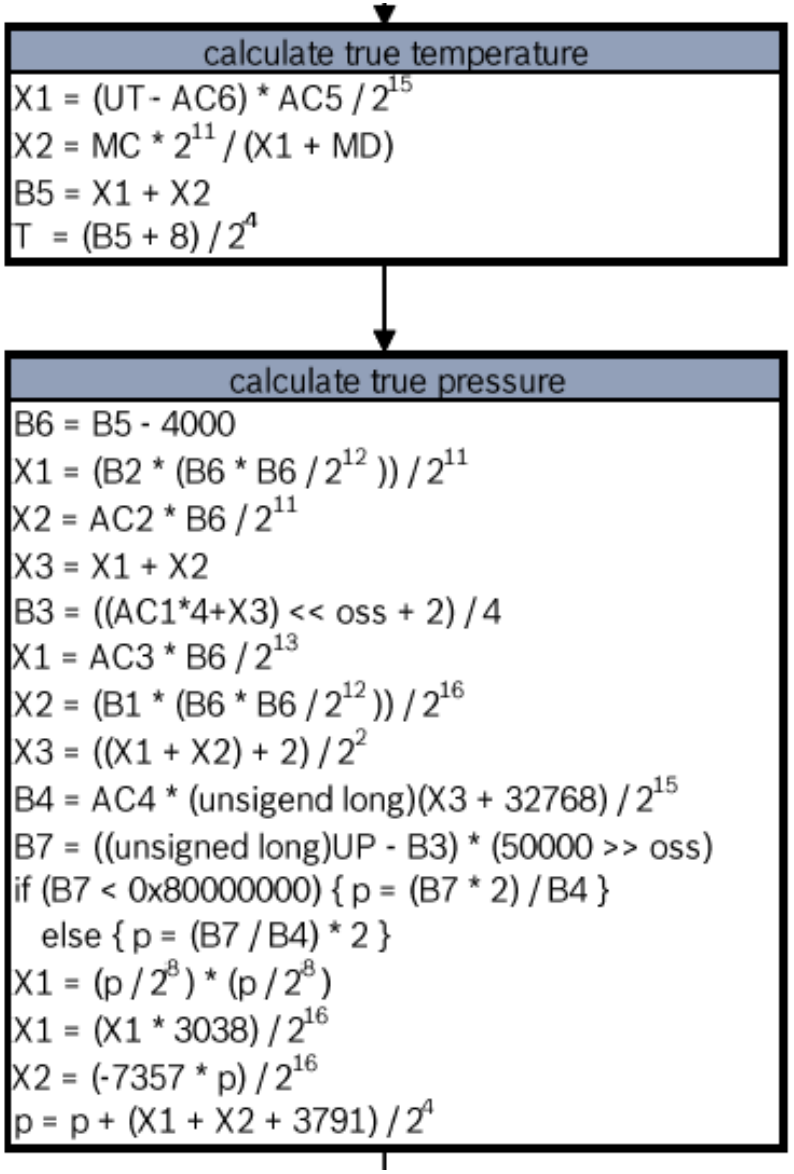
Nyomásmérés: 300 – 1100 hPa (9000 - -300m)
Tápfeszültség: 1,8 – 3,6 V
Áramfelvétel: 5 µA (1 mintavétel/s esetén)
Kis zaj: 0.06hPa (0.5m) kisfogyasztású mód
0.02hPa (0.17m) nagyfelbontású mód
Jellemzők: Hőmérő, I2C felület, gyárilag kalibrált

A kiszámítás menete

Table 5: Calibration coefficients

	BMP180 reg adr	
Parameter	**MSB**	**LSB**
AC1	0xAA	0xAB
AC2	0xAC	0xAD
AC3	0xAE	0xAF
AC4	0xB0	0xB1
AC5	0xB2	0xB3
AC6	0xB4	0xB5
B1	0xB6	0xB7
B2	0xB8	0xB9
MB	0xBA	0xBB
MC	0xBC	0xBD
MD	0xBE	0xBF

UT és **UP** a nyers hőmérséklet és nyomás adat
oss – oversampling (0 – 3)



Helyi légnyomás átszámítása tengerszintre

- Ami mérünk, az a ***p*** helyi légnyomás, ami a tengerszint feletti helyi magasság és az időjárási viszonyok függvénye
- Az időjárási adatoknál a mért légnyomást a tengerszintre átszámított p_0 légnyomást szokás megadni, az átszámítás az alábbi képlettel végezhető (ahol ***altitude*** a méterben mért tengerszint feletti helyi magasság)

$$p_0 = \frac{p}{\left(1 - \frac{\text{altitude}}{44330}\right)^{5.255}}$$

- Például Debrecen magasságát 120 m-nek véve, azt kapjuk, hogy

$$p_0 \approx p \cdot 1{,}01434635$$

- Még durvább közelítéssel Debrecenben $p_0 \approx p + 1440\,Pa$

A bmp180.py programkönyvtár

- A github.com/shaoziyang/microbit-lib gyűjteményben található **bmp180.py** programkönyvtárat használtuk
- Példányosítás: b = bmp180.BMP180()
  Az **I2C** kivezetések kötöttek: SCL = pin19, SDA = pin20
- Metódusok:
  - **get()** – egy mérési ciklus végrehajtása és az eredmény kiszámítása
  - **Temperature()** – a mért hőmérséklet kiolvasása (°C)
  - **Pressure()** – a helyi légnyomás kiolvasása (Pa)
  - **Altitude()** – az „egyezményes nyomásmagasság” kiszámítása, tegerszinti nyomásként az 1013.25 hPa értéket használva

Megjegyzés: az ilyen módon meghatározott „magasság” az időjárási viszonyok függvénye, a repülésben csak nagy magasságokban (8-10 000 láb fölött) használatos, ahol 2-300 m-es eltérés nem okoz gondot, a *minden repülő azonosan számítja* a magasságot…

micro:bit

Reference

Ideas

API

Project

Variables
Keep track of data that…

Display
The micro:bit's LED display…

Buttons
Use button inputs in your…

Loops
Count and repeat sets of…

Logic
Making decisions in code

Accelerometer
Detect gestures and…

BME180_serial

```
1  from microbit import *
2
3  import bmp180
4
5  b = bmp180.BMP180()
6
7  while True:
8      sleep(500)
9      b.get()
10     print('Temp = ',round(b.Temperature(),1),' C')
11     sleep(2000)
12     p = (b.Pressure()+1440)/100.0
13     print('Pressure = ',round(p,1),' hPa')
14     sleep(2000)
15
```

micro:bit flashed ✓ Hide serial

```
Temp =  24.5  C
Pressure =  1014.6  hPa
Temp =  24.5  C
Pressure =  1014.5  hPa
Temp =  24.5  C
Pressure =  1014.4  hPa
Temp =  24.5  C
Pressure =  1014.5  hPa
Temp =  24.6  C
```

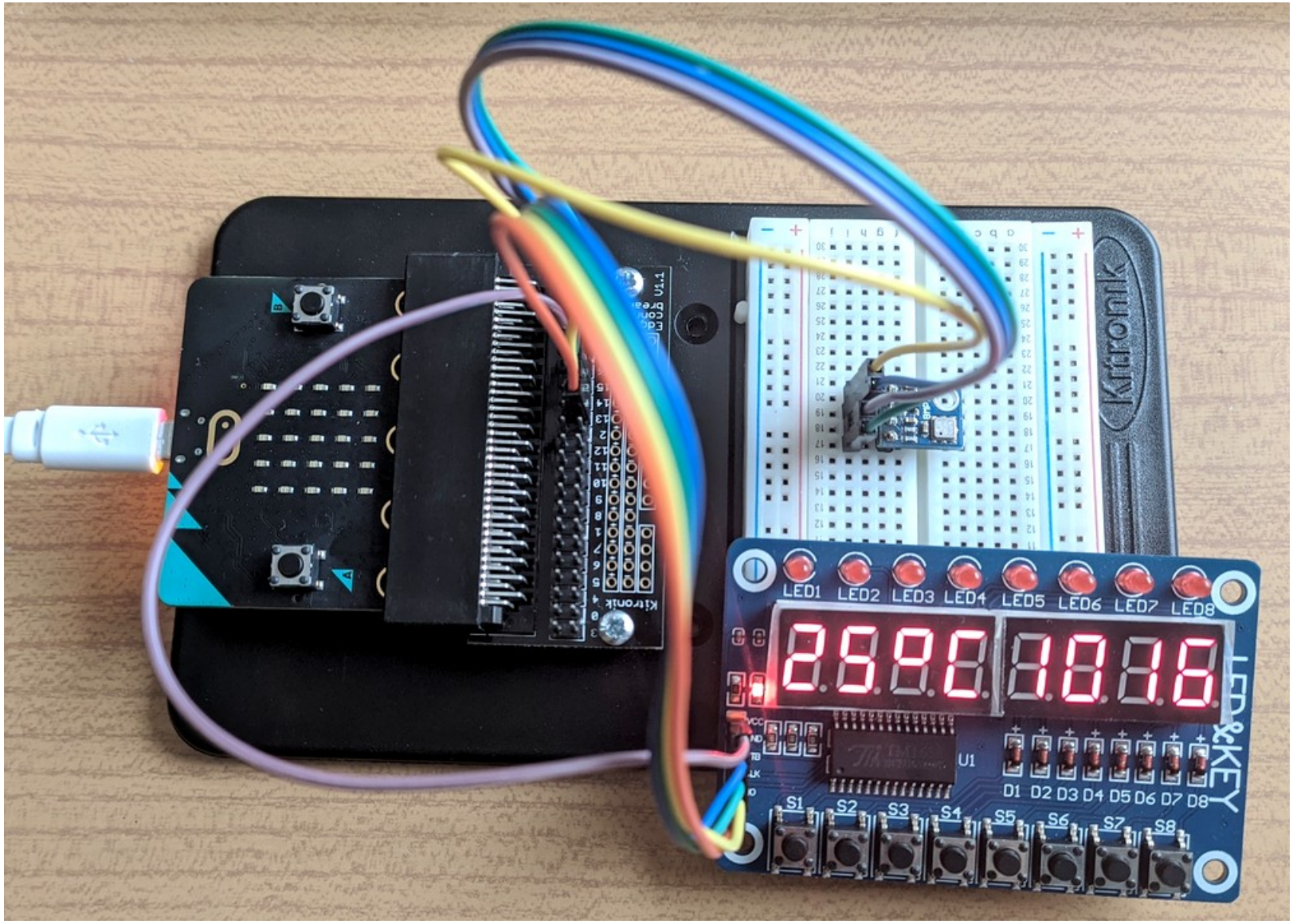
BME180_tm1638.py

- Kombináljuk össze a **bmp180** és a **tm1638** könyvtárat és a hétszegmenses LED kijelzőn írassuk ki a légnyomás és a hőmérséklet értékét!

```
from microbit import *
import bmp180
from tm1638 import TM1638
tm = TM1638(stb=pin15, clk=pin14, dio=pin13)
b = bmp180.BMP180()

tm.clear()          # kijelő törlése

while True:
    sleep(5000)
    b.get()         # mérés
    p = (b.Pressure()*1.01434635)/100.0
    tm.number(int(p+0.5))     # jobbra igazított kiíratás
    tm.temperature(int(b.Temperature()+0.5),0) # balra igazított
                                               # kiíratás
```

Korrekció 120 m magasságra

BME180_tm1638.py



Fájlok hozzáadása a Mu editorban