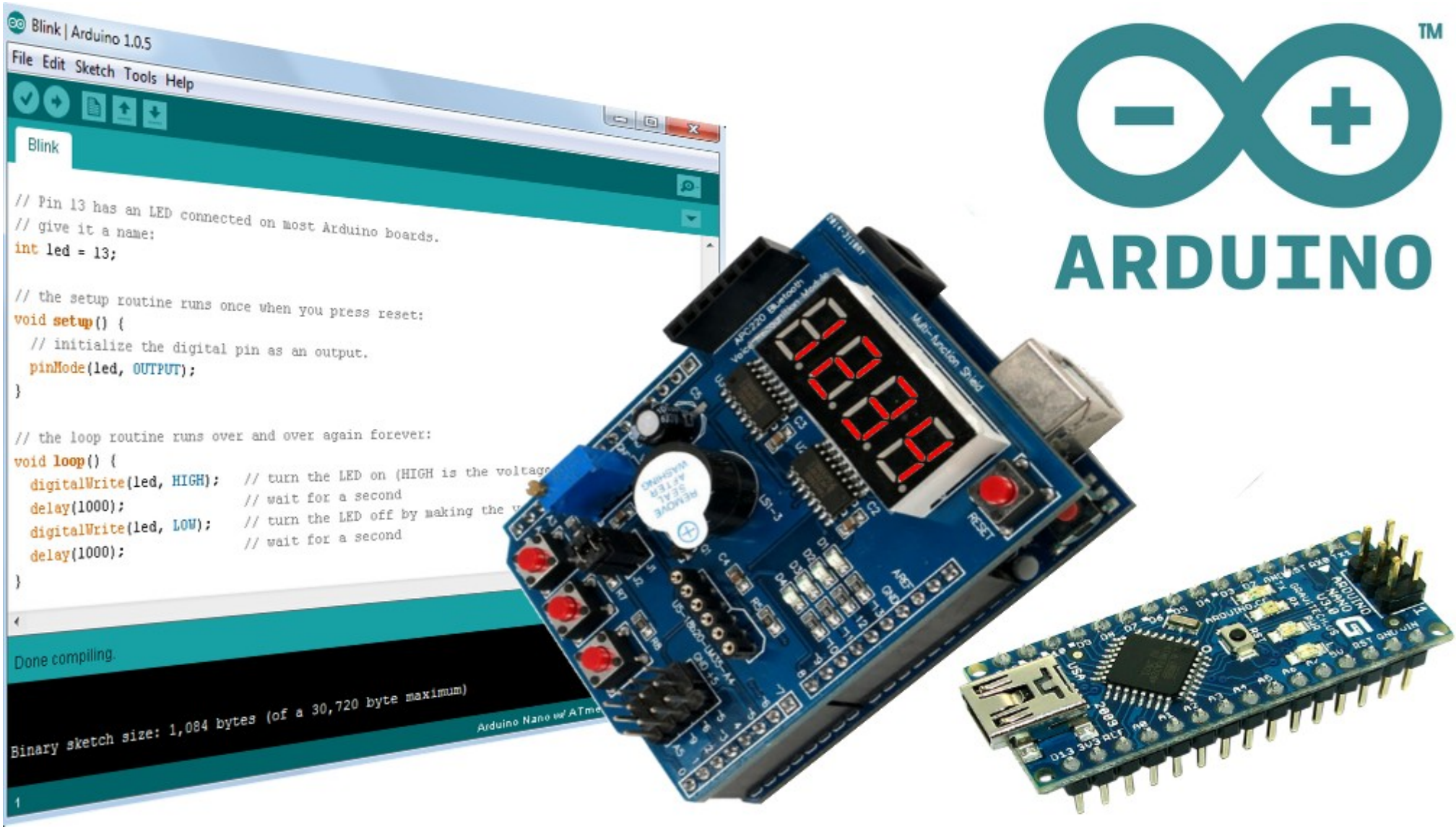
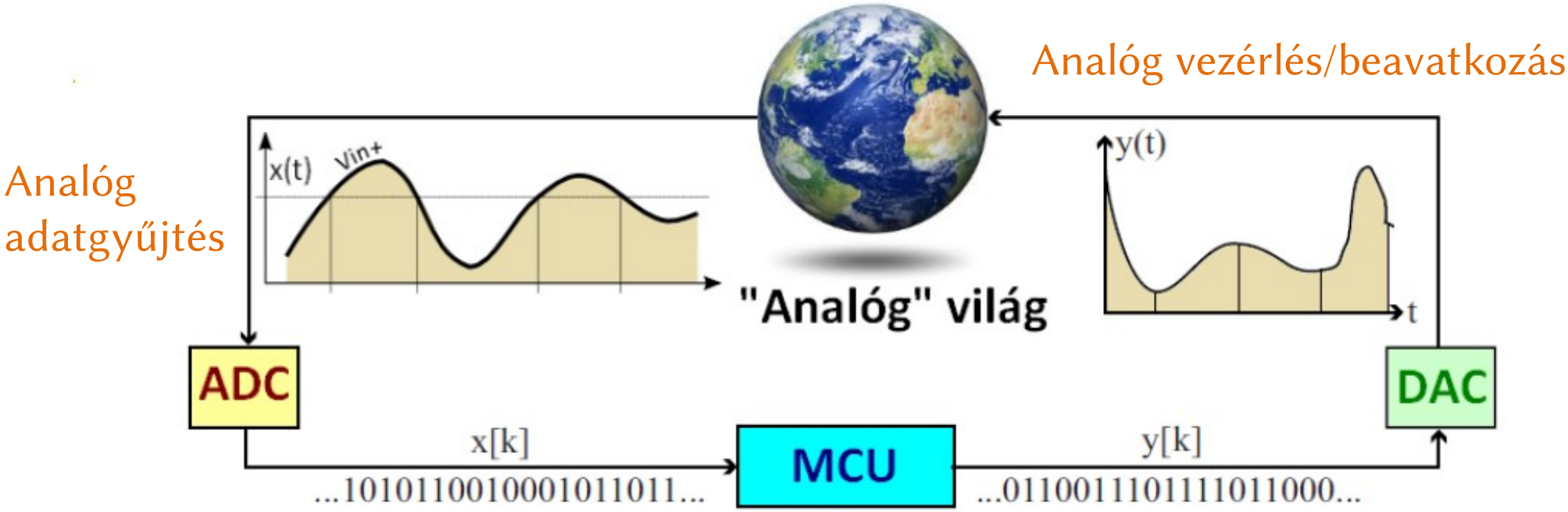
Arduino tanfolyam kezdőknek és haladóknak

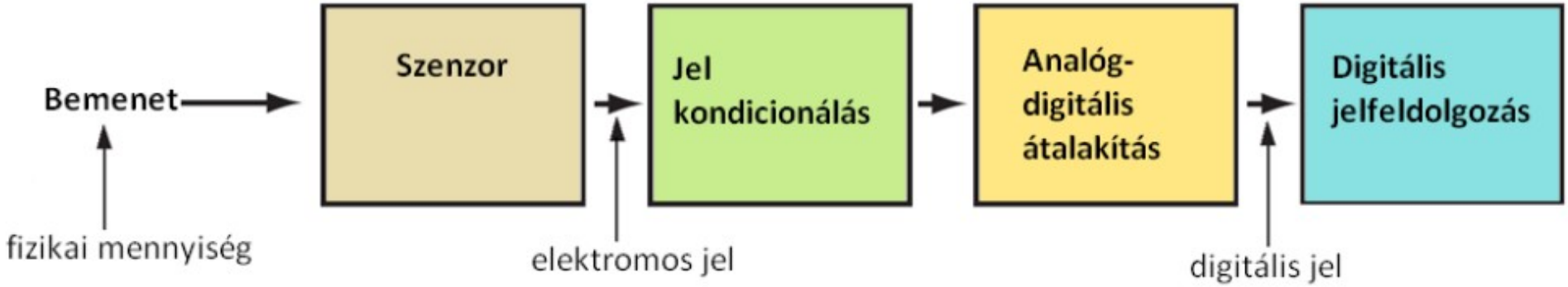


4. Az analóg-digitális átalakító (ADC) használata

Analóg jelfeldolgozás

- Analóg világban élünk, de digitális mikrovezérlővel dolgozunk…





Az analóg adatgyűjtő ág elemei

- **Szenzor:**
  - A folytonos fizikai mennyiségeket (pl. hőmérséklet, nyomás, páratartalom, sebesség, áramlási sebesség, elmozdulás, gyorsulás, szöggyorsulás) elektromos jellé alakítja (feszültséggé vagy árammá).
- **Jel kondicionálás (szűrés, erősítés, stb.):**
  - A mérendő mennyiség elektromos jellé alakítása után még szűrésre, jel erősítésre, impedancia illesztésre is szükség lehet, hogy az analóg-digitális átalakító (ADC) bemeneti tartományába transzformáljuk az átalakítandó jelet.
- **Analóg-Digitális Átalakító (ADC):**
  - *Bemenet:* a mérendő jel
  - *Kimenet:* a mérendő jellel arányos számot reprezentáló digitális kód

Analóg–digitális átalakító (ADC)

- Az ADC feladata az, hogy diszkrét kódokká alakítsa a bejövő jelet
- A konverzió digitális értéke (N_{ADC}):
  - Végkitérés: N_{ADC}= 1023, ha a bemenő jel ≥ V_{R+} –1.5*LSB
  - Nulla: N_{ADC} = 0, ha a bemenő jel ≤ V_{R-} + 0.5 LSB
  - Közbeeső értékekre:

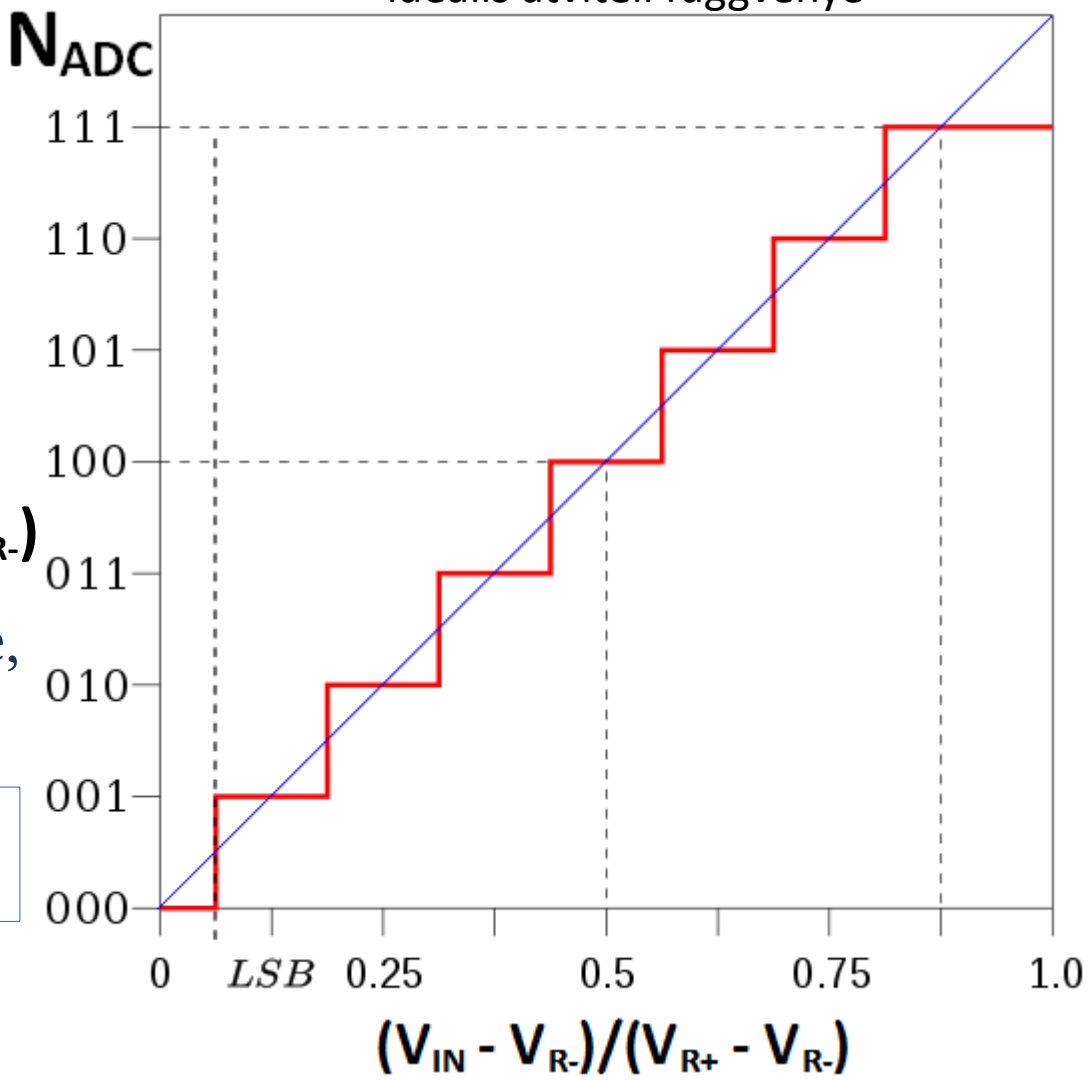
$$N_{ADC} = 1024 * (V_{IN} - V_{R-})/(V_{R+} - V_{R-})$$

- A fenti képletből V_{IN}-t kifejezve, ezt kapjuk:

$$V_{IN} = (V_{R+} - V_{R-}) * N_{ADC}/1024 + V_{R-}$$

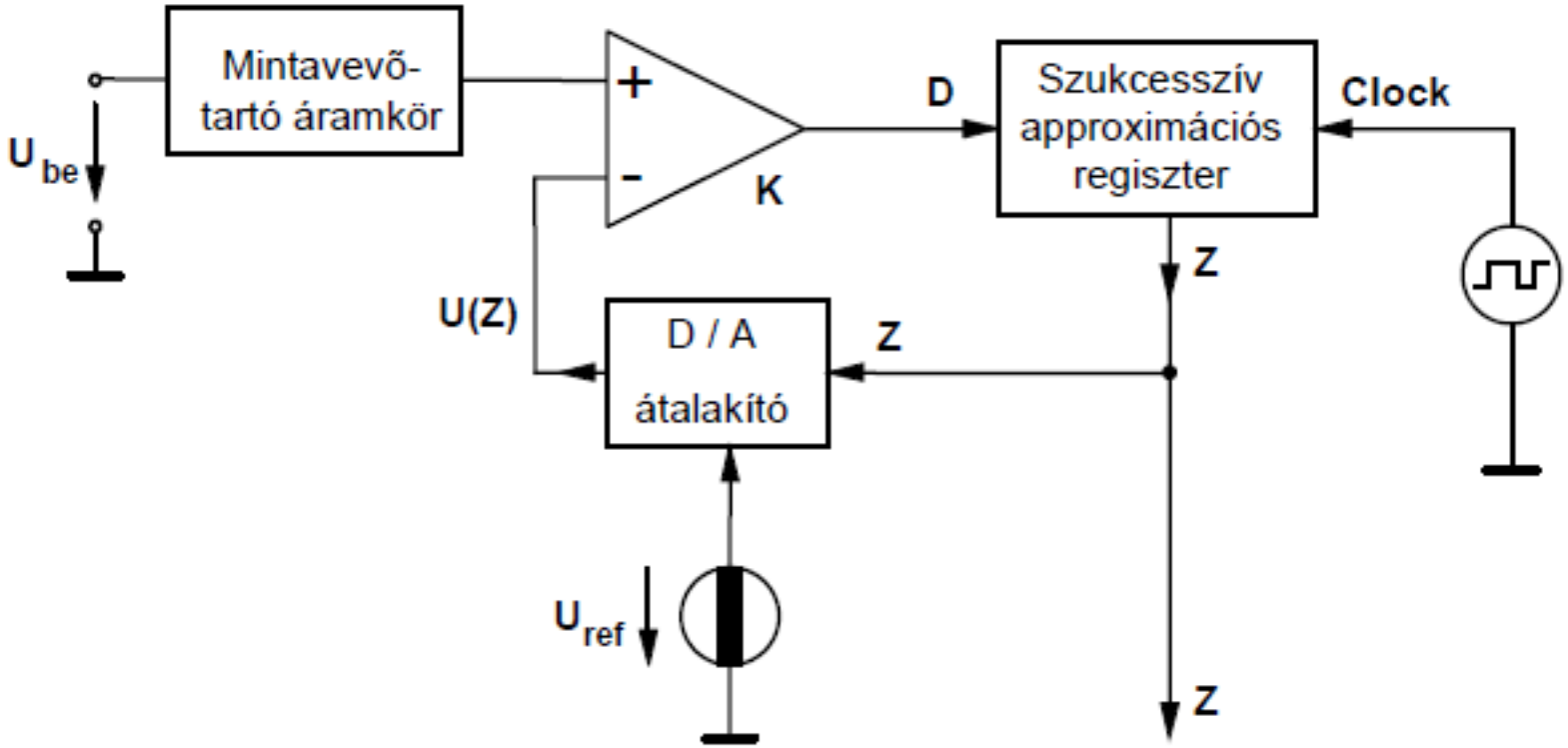
- V_{R-} általában = 0

Egy 3-bites átalakító ideális átviteli függvénye



Szukszesszív approximációs ADC

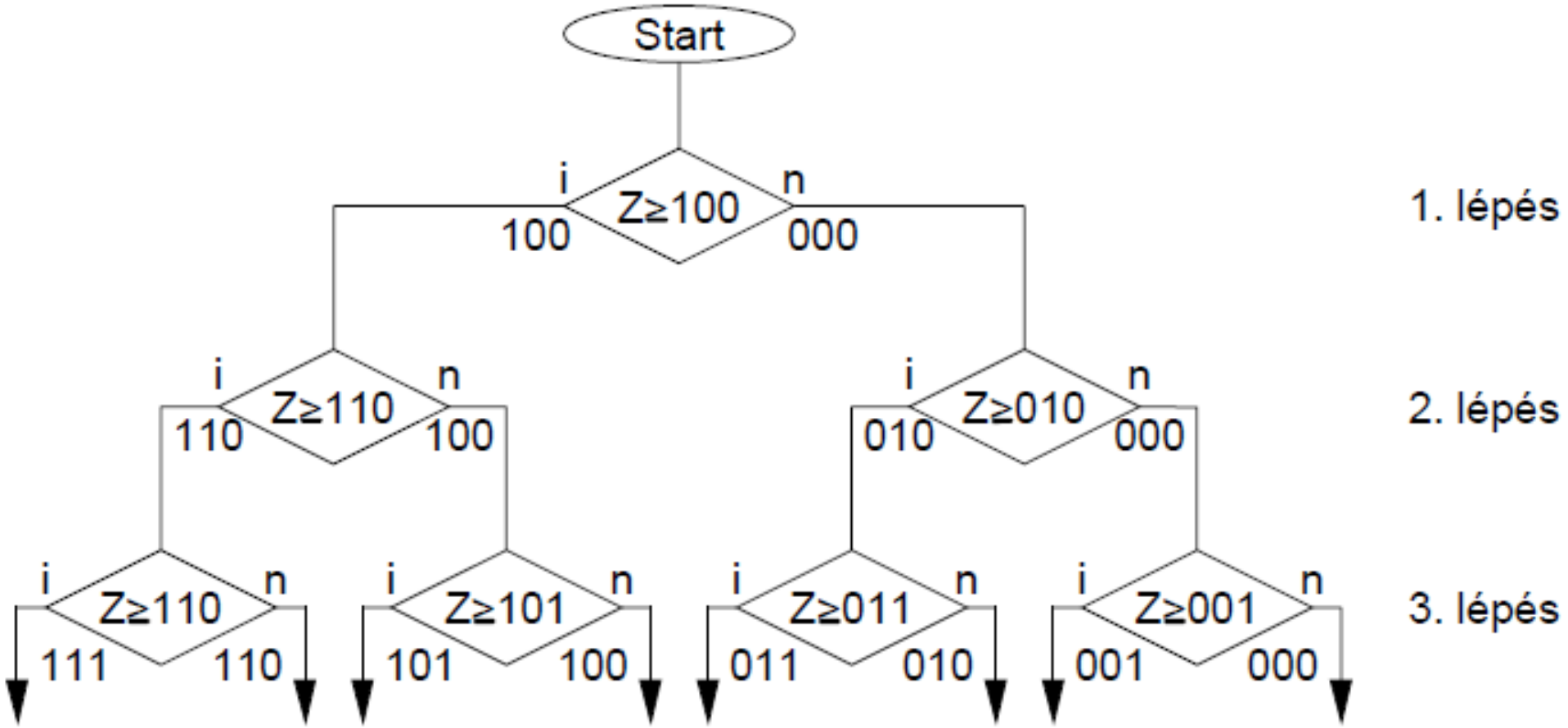
- A szukszesszív approximáció fokozatos megközelítést jelent
- Első lépésben a fél feszültséggel hasonlítjuk össze a mintavételezett jelet, s a döntéstől függően a Z regiszter MSB bitje 1 lesz, vagy 0
- A többi bitet további döntési lépésekben kapjuk meg



Forrás: BME Irányítástecnika I. (segédlet)

Szukszesszív approximációs ADC

- Az ábrán egy 3 bites A/D átalakító konverziós lépéseinek döntési sorozata látható (az Arduino ADC-je 10 bites)



Forrás: BME Irányítástecnika I. (segédlet)

Az ADC-t kezelő függvények

- **analogReference(*típus*)** – az analóg bemenetek viszonyítási (referencia) feszültségét konfigurálhatjuk vele
- A választható opciók:
  `DEFAULT` – a tápfeszültség a referencia (5V helyett inkább 4,75 V)
  `INTERNAL` – a belső 1,1 V-os referencia
  `EXTERNAL` – külső forrásból a $\mathbf{V_{ref}}$ lábra adhatunk feszültséget (0-5V)
- **analogRead(*pin*)** – elindít egy mérést a megadott analóg bemeneten (`A0-A7`) és a visszatérési érték a konverzió eredménye lesz (0 – 1023 közötti egész szám)

```
void setup() {
  analogReference(DEFAULT);      // Vcc a referencia
  Serial.begin(9600);            // Soros port konfigurálása
}
void loop() {
  int val = analogRead(A0);      // Az A0 bemenet feszültségét mérjük
  Serial.println(val);           // Kiíratjuk az eredményt
}
```

adc_potmeter.ino

- Mérjük meg a potméterrel leosztott feszültséget és írassuk ki!

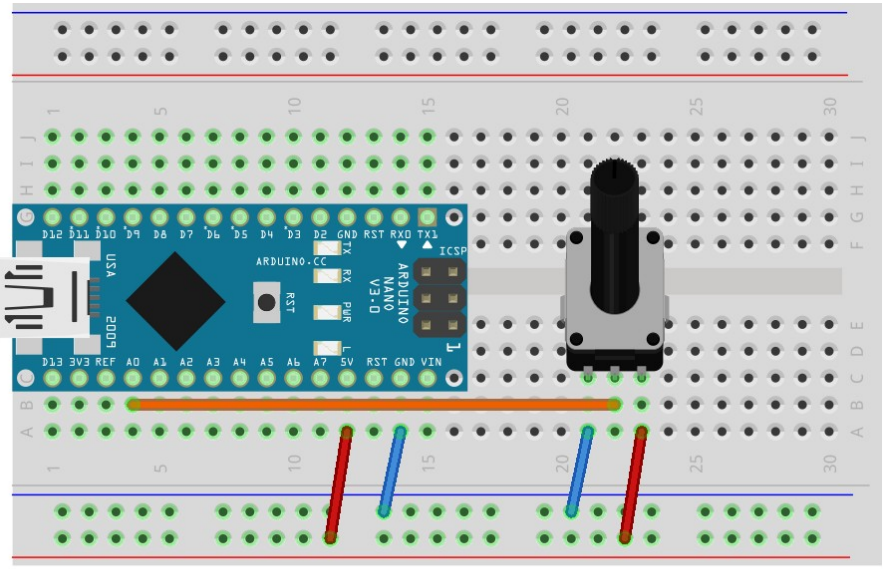
```
void setup() {
  Serial.begin(9600);
  analogReference(DEFAULT);
  Serial.println("ADC potmeter reading");
}

void loop() {
  int reading = analogRead (A0);
  float voltage=(reading*4.75)/1024.0;
  Serial.print("ADC: ");
  Serial.print(reading);
  Serial.print("  Voltage: ");
  Serial.print(voltage, 2);
  Serial.println(" V");
  delay(2000);
}
```

VCC a referencia

A0 a bemenet

VCC tényleges értéke



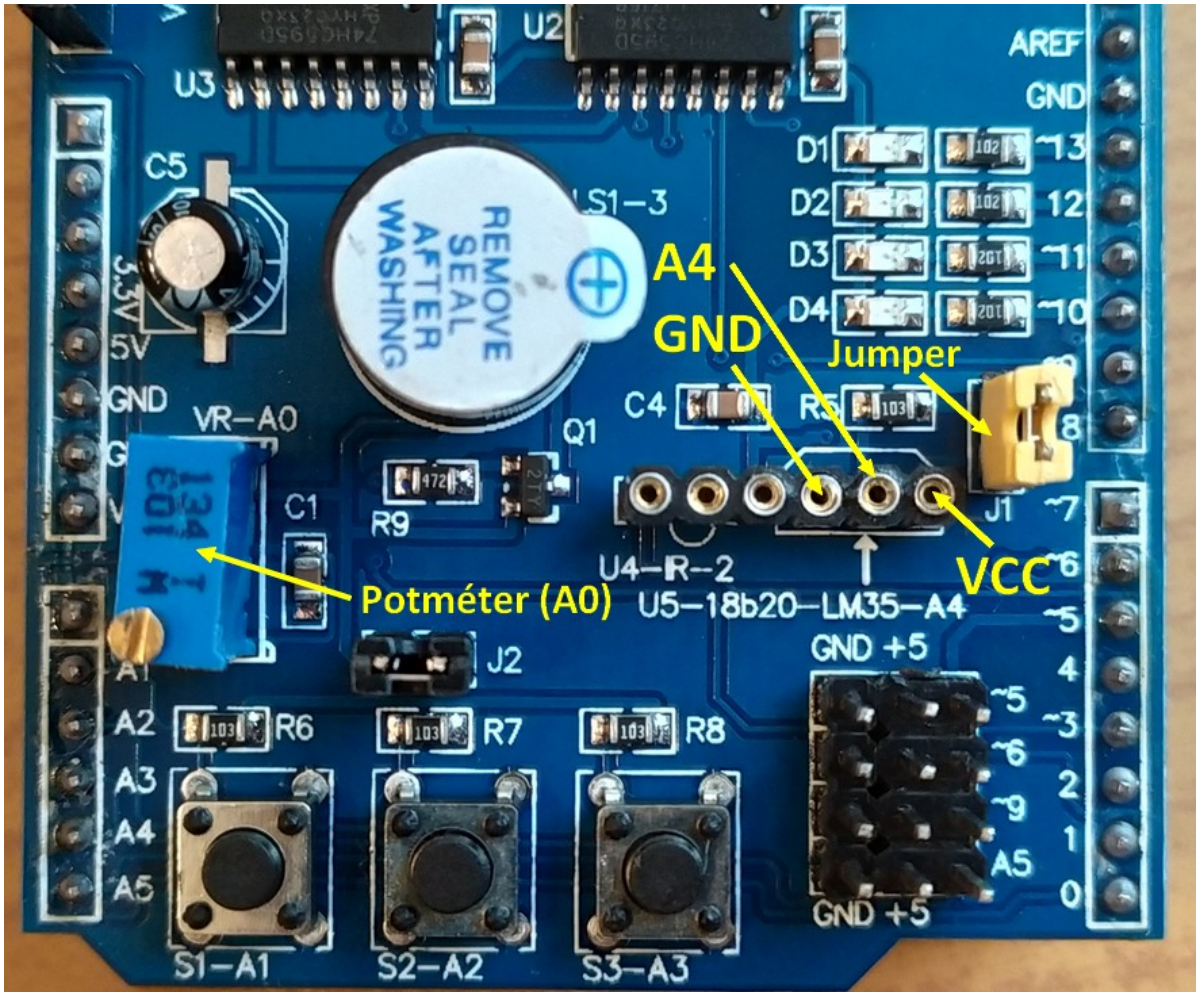
fritzing

```
COM5

ADC potmeter reading
ADC: 551  Voltage: 2.53 V
ADC: 550  Voltage: 2.53 V
ADC: 550  Voltage: 2.53 V
ADC: 550  Voltage: 2.53 V
ADC: 550  Voltage: 2.53 V
ADC: 550  Voltage: 2.53 V
ADC: 550  Voltage: 2.53 V
ADC: 550  Voltage: 2.53 V
ADC: 550  Voltage: 2.53 V
ADC: 550  Voltage: 2.53 V
```

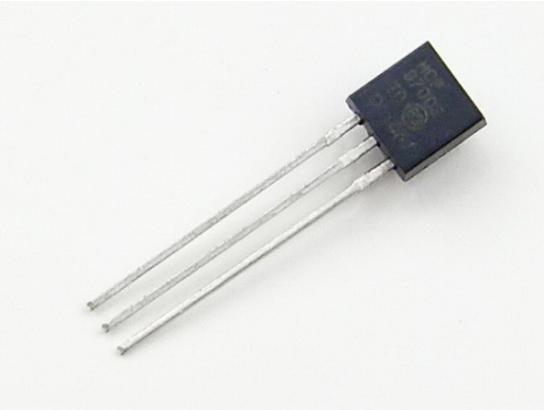
Analóg perifériák a Multifunkciós kártyán

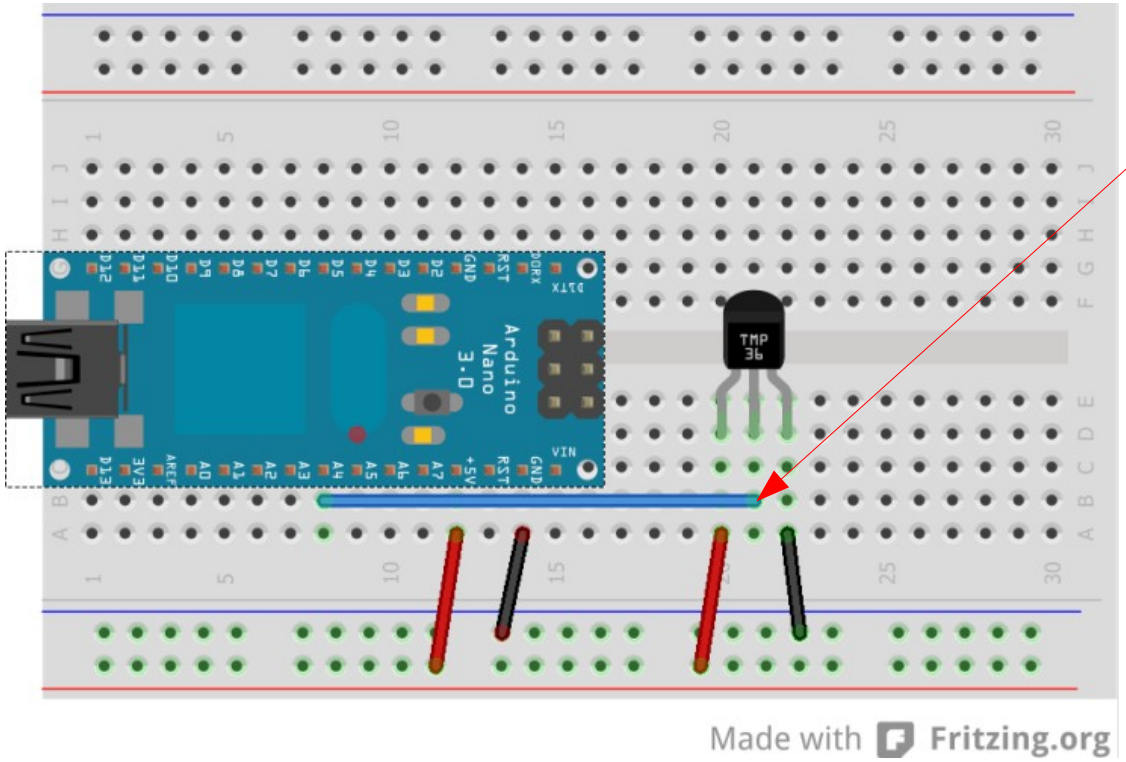
- A csavarhúzóval állítható potméter a tápfeszültséget osztja le, csúszkája az **A0** bemenetre van kötve
- A csatlakozó hüvelysor jobboldali három pontja egy analóg (**LM35, MCP9700**), vagy egy digitális (**DS18B20**) hőmérő fogadására alkalmas
- A hőmérő kimenete az **A4** lábra csatlakozik
- A **J1** átkötés 10 kΩ-mal a tápfeszültségre köti az **A4** bemenetet, ami csak a **DS18B20** esetén kell



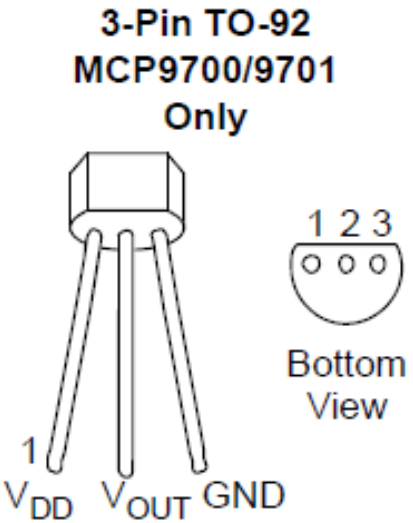
Hőmérséklet mérése analóg hőmérővel

- Microchip **MCP9700**
  - VDD = 2,5 – 5,5 V
  - Mérési tart.: –40 – 150 °C
  - Érzékenység: 10 mV / °C
  - Nullapont: 500 mV @ 0 °C





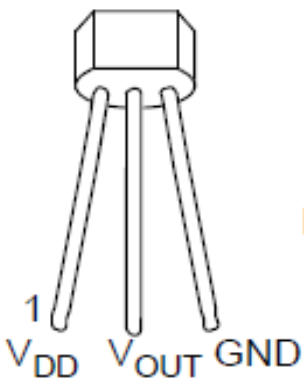
Most az **A4** bemenetre kötöttük a hőmérőt, de más bemenetet is használhatunk!

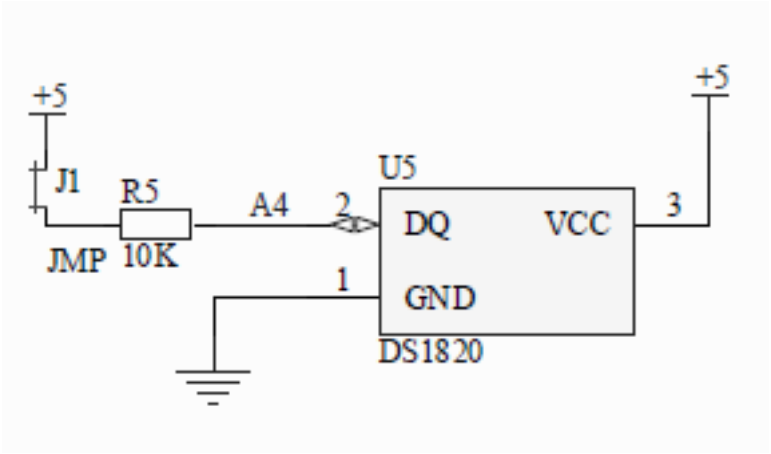


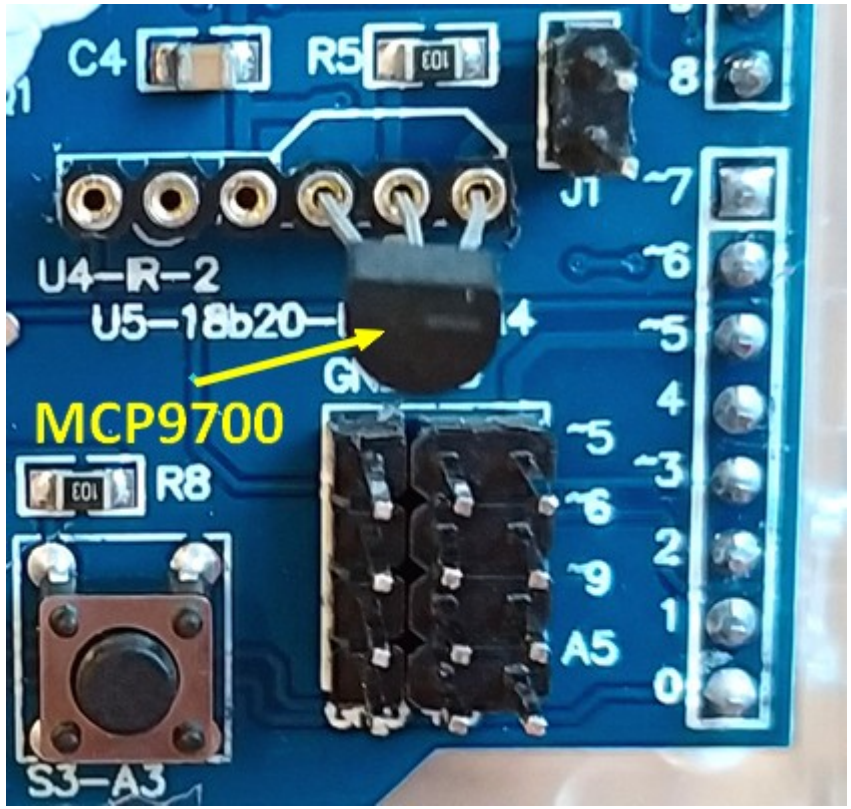
MCP9700 a Multifunkciós kártyán

- Az **MCP9700** hőmérőt a homloklapjával fölfelé, azaz a nyomtatott áramkörre felfestett jelzéshez képest 180 fokkal elforgatva kell csatlakoztatni
- A hőmérő kimenete itt is az **A4** analóg bemenetre csatlakozik, tehát a kapcsolás megegyezik az előző oldalon bemutatottal
- A **J1** átkötést ne felejtsük el eltávolítani!

MCP9700







Kijelzés a multifunkciós kártyán

- Jelenítsük meg az **ADC** konverzió eredményét a Multifunkciós kártya hétszegmenses kijelzőjén, a múlt órán tanultak segítségével!
- A számjegyek előállítása az egészosztás és a maradékos osztás segítségével végezhető
- Legyen például PotValue az ADC konverzió eredménye, ezt az alábbi módon írathatjuk ki:

```
WriteNumberToSegment(0 ,  PotValue / 1000);
WriteNumberToSegment(1 , (PotValue / 100) % 10);
WriteNumberToSegment(2 , (PotValue / 10) % 10);
WriteNumberToSegment(3 ,  PotValue % 10);
```

- Ne feledjük el, hogy a multiplex kijelzés miatt a megjelenítést végtelen ciklusban, folyamatosan ismételgetni kell!

adc_display.ino

```
/* A hétszegmenses kijelzőt vezérlő shift regiszter lábai */
#define LATCH_DIO 4          // Latch regiszter vezérlőjel
#define CLK_DIO   7          // Órajel kimenet
#define DATA_DIO  8          // Soros adatkimenet

#define Pot1      A0         // Potméter A0-ra csatlakozik

/* Számjegyek (0 - 9) szegmensképe, negatív logikával */
const byte SEGMENT_MAP[]={0xC0,0xF9,0xA4,0xB0,0x99,0x92,0x82,0xF8,0x80,0x90};
/* Számjegy (1 - 4) kiválasztó jelek */
const byte SEGMENT_SELECT[] = {0xF1, 0xF2, 0xF4, 0xF8};
long nexttime;               // következő időpont megjegyzése

void setup () {
  Serial.begin(115200);      // Soros port inicializálása
  analogReference(DEFAULT); // VCC a referencia
  /* A DIO kivezetések kimenetre állítása */
  pinMode(LATCH_DIO, OUTPUT);
  pinMode(CLK_DIO, OUTPUT);
  pinMode(DATA_DIO, OUTPUT);
  nexttime = 1000 + millis();
}
```

adc_display.ino

```
void loop() {
  int PotValue;
  if (millis() > nexttime) {                                    TASK1
    PotValue = analogRead(Pot1);
    Serial.print("Potentiometer: ");
    Serial.println(PotValue);
    nexttime += 1000;                  // Mérés és kijelzés 1 másodpercenként
  }

  /* Mérési szünetekben pörgetjük a kijelzést */
  WriteNumberToSegment(0 , PotValue / 1000);
  WriteNumberToSegment(1 , (PotValue / 100) % 10);          TASK2
  WriteNumberToSegment(2 , (PotValue / 10) % 10);
  WriteNumberToSegment(3 , PotValue % 10);
}

/* A soron következő számjegy megjelenítése */
void WriteNumberToSegment(byte Segment, byte Value) {
  digitalWrite(LATCH_DIO, LOW);
  shiftOut(DATA_DIO, CLK_DIO, MSBFIRST, SEGMENT_MAP[Value]);
  shiftOut(DATA_DIO, CLK_DIO, MSBFIRST, SEGMENT_SELECT[Segment] );
  digitalWrite(LATCH_DIO, HIGH);
}
```

adc_display.ino

- Az ábrán a potméterrel fél tápfeszültségre beállított feszültség esetén az ADC által visszaadott szám: 512
- Tápfeszültség: 4.90 V
- A potméter csúszkáján mért feszültség: 2.45 V

AnalogThermometer.ino

```
void setup () {
   Serial.begin(9600);
   analogReference(INTERNAL);
   Serial.println("Analóg hőmérő");
}

void loop () {
   long reading = analogRead (A4);
//--- Átszámítjuk mV-ba
   long voltage = reading*1100/1024;
   Serial.print (voltage);
   Serial.print (" mV, ");
//--- Átszámítjuk Celsius fokokra
   float tempC = (voltage - 500)/10.0;
   Serial.print (tempC,1);
   Serial.print (" °C, ");
//--- Celsiusból Fahrenheit fokokba
   float tempF = (tempC * 9/5) + 32;
   Serial.print (tempF,1);
   Serial.println (" °F");
   delay (5000);
}
```

Vref = 1,1 V

Hőmérés **MCP9700A** hőmérővel

A hőmérőt az **A4** bemenetre kötöttük

Analóg hőmérő
763 mV, 26.3 °C, 79.3 °F
771 mV, 27.1 °C, 80.8 °F
770 mV, 27.0 °C, 80.6 °F
770 mV, 27.0 °C, 80.6 °F
771 mV, 27.1 °C, 80.8 °F
770 mV, 27.0 °C, 80.6 °F
770 mV, 27.0 °C, 80.6 °F
771 mV, 27.1 °C, 80.8 °F
770 mV, 27.0 °C, 80.6 °F
768 mV, 26.8 °C, 80.2 °F
779 mV, 27.9 °C, 82.2 °F

AnalogThermometer2.ino

```
void setup () {
  Serial.begin(9600);
  analogReference(INTERNAL);
  Serial.println("Analóg hőmérő átlagolással");
}

void loop () {
  long reading = 0;
  for (int i = 0; i < 1100; i++) {
    reading += analogRead(A4);
  }
  long voltage = reading/1024;
  Serial.print (voltage);
  Serial.print (" mV, ");
  float tempC = (voltage-500)/10.0;
  Serial.print (tempC, 1);
  Serial.print (" °C, ");
  float tempF = (tempC*9/5)+32;
  Serial.print (tempF, 1);
  Serial.println (" °F");
  delay (5000);
}
```

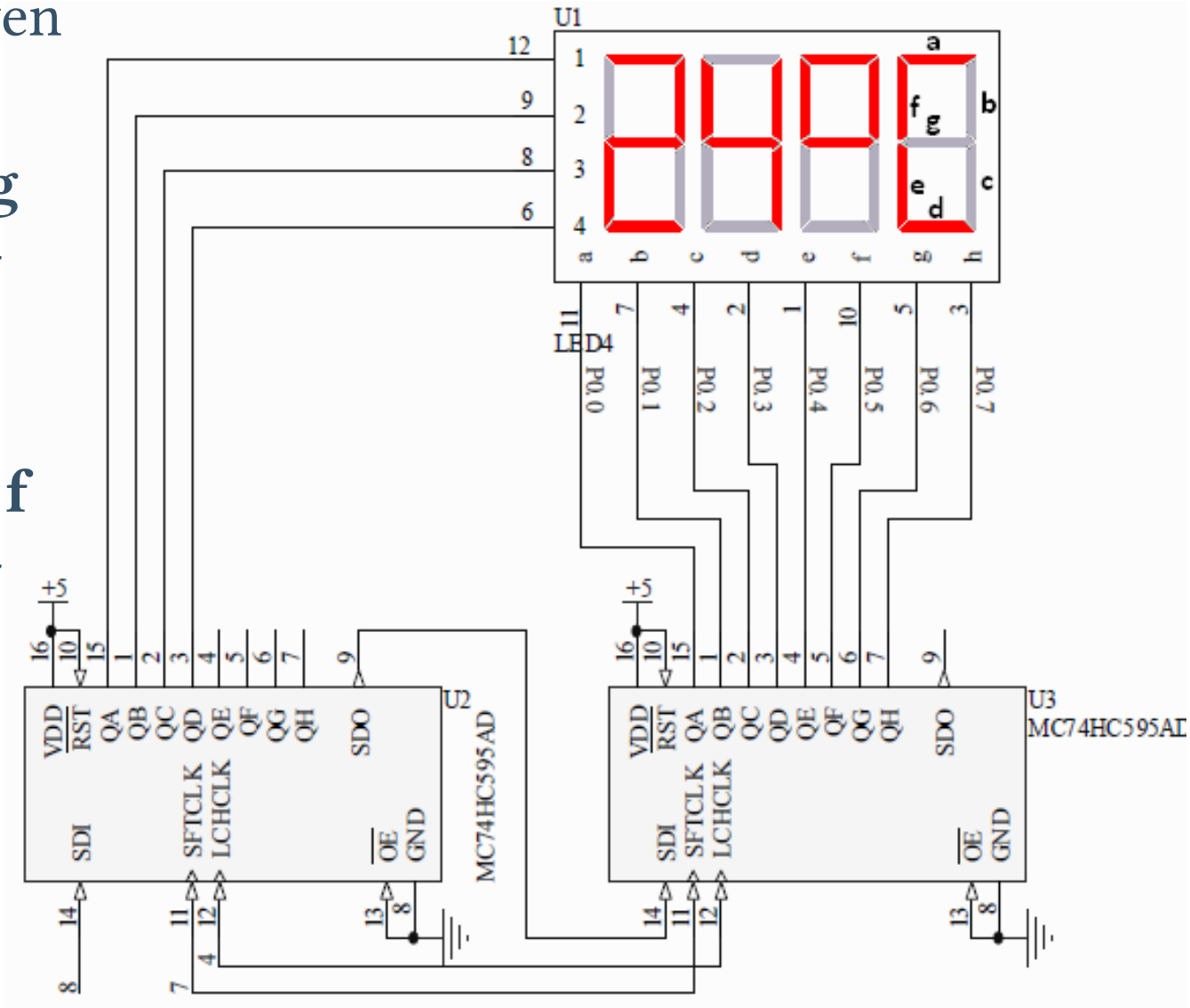
Hőmérés **MCP9700A** hőmérővel, átlagolással

1100 db mérés eredményét összeadjuk és nem szorzunk 1100-za!

Analóg hőmérő átlagolással
772 mV, 27.2 °C, 81.0 °F
771 mV, 27.1 °C, 80.8 °F
772 mV, 27.2 °C, 81.0 °F
772 mV, 27.2 °C, 81.0 °F
771 mV, 27.1 °C, 80.8 °F
771 mV, 27.1 °C, 80.8 °F
771 mV, 27.1 °C, 80.8 °F
771 mV, 27.1 °C, 80.8 °F
771 mV, 27.1 °C, 80.8 °F
771 mV, 27.1 °C, 80.8 °F

Hőmérséklet kijelzése a multifunkciós kártyán

- Készítsünk szobahőmérőt a Multifunkciós kártya felhasználásával!
- Az utolsó két számjegyen mindig a °C lesz kiírva
- A fok jelhez az **a, b, f, g** szegmenseket kell kivilágítani: 0b1001 1100, azaz **0x9C**
- A C betűhöz az **a, d, e, f** szegmenseket kell kivilágítani: 0b1100 0110, azaz **0xC6**
- Az első két jegyen a hőmérsékletet írjuk ki, Celsius fokokban



thermometer_display.ino 3/1

```
/* A hétszegmenses kijelzőt vezérlő shift regiszter lábai */
#define LATCH_DIO 4          // Latch regiszter vezérlőjel
#define CLK_DIO   7          // Órajel kimenet
#define DATA_DIO  8          // Soros adatkimenet
/* Számjegyek (0 - 9) szegmensképe, negatív logikával */
const byte SEGMENT_MAP[]={0xC0,0xF9,0xA4,0xB0,0x99,0x92,0x82,0xF8,0x80,0x90};
/* Számjegy (1 - 4) kiválasztó jelek */
const byte SEGMENT_SELECT[] = {0xF1, 0xF2, 0xF4, 0xF8};

byte segment_data[] = {0xFF, 0xFF, 0x9C, 0xC6}; // A frissítendő szegmensképe
byte counter = 50;           // Számláló a mérések átlagolásához
byte idx = 0;                // Futó index a kijelzéshez
long nexttime;               // következő időpont megjegyzése
long sum = 0;                // Ebben összegezzük a mért adatokat

void setup () {
  Serial.begin(115200);
  /* Set DIO pins to outputs */
  pinMode(LATCH_DIO, OUTPUT);
  pinMode(CLK_DIO, OUTPUT);
  pinMode(DATA_DIO, OUTPUT);
  nexttime = millis();
}
```

thermometer_display.ino 3/2

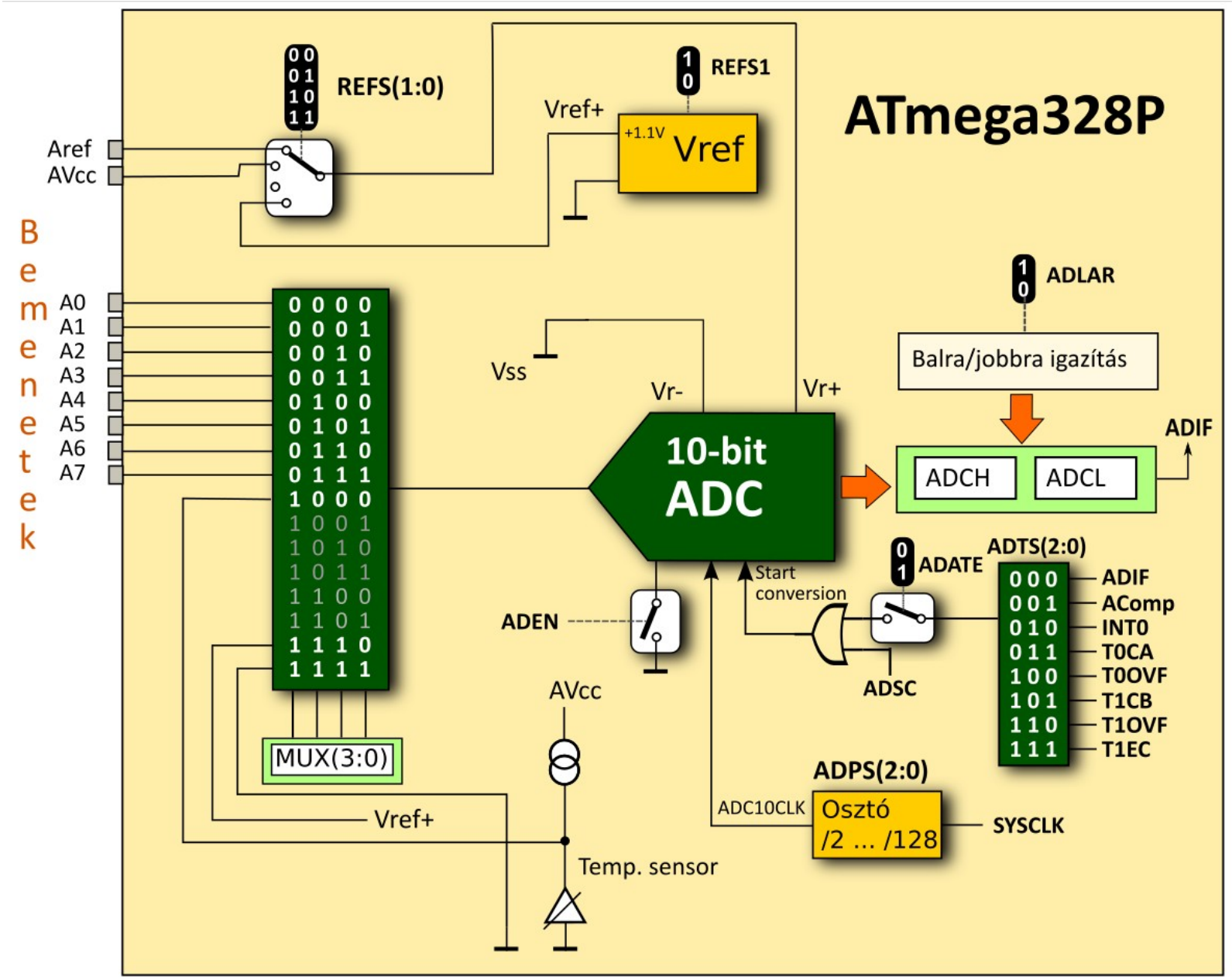
```
void loop() {
  long voltage, tempC;
  if (millis() > nexttime) {           // TASK1 100 ms-onként fut le
    sum += analogRead(A4);
    if (--counter == 0) {
      voltage = sum * 100 / 1024;     // Feszültség mV-ban
      tempC = (voltage - 500) / 10;   // Hőmérséklet Celsius fokokban
      sum = 0;
      counter = 50;
      segment_data[0] = SEGMENT_MAP[tempC / 10];
      segment_data[1] = SEGMENT_MAP[tempC % 10];
      float temp = (voltage - 500) / 10.0;
      Serial.print("Voltage: ");
      Serial.print(voltage);
      Serial.print(" mV  Temp: ");
      Serial.print(temp,1);
      Serial.println(" °C");
    }
    nexttime += 100;
  }
  /* Mérési szünetekben pörgetjük a kijelzést (TASK2) */
  refresh_display();
}
```

thermometer_display.ino 3/3

```
/* A soron következő számjegy megjelenítése */
void refresh_display() {
  byte i;
  i = idx++ & 0x3;
  digitalWrite(LATCH_DIO, LOW);
  shiftOut(DATA_DIO, CLK_DIO, MSBFIRST, segment_data[i]);
  shiftOut(DATA_DIO, CLK_DIO, MSBFIRST, SEGMENT_SELECT[i] );
  digitalWrite(LATCH_DIO, HIGH);
}
```

- A `refresh_display()` függvényben egy meghívás alkalmával csak egy számjegy kijelzését frissítjük, hogy minél rövidebb ideig tartsuk fel a program futását
- Az **idx** globális változó alsó két bitje tartja nyilván, hogy éppen melyik számjegy frissítése következik

Az ADC blokkvázlata



Az ADC regiszterei

- **ADMUX** – referenciafeszültség és bemeneti csatorna választó

Bit	7	6	5	4	3	2	1	0	
(0x7C)	REFS1	REFS0	ADLAR	–	MUX3	MUX2	MUX1	MUX0	ADMUX
Read/Write	R/W	R/W	R/W	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

REFS – 00: `EXTERNAL`, 01: `DEFAULT` (Avcc), 11: `INTERNAL` (1,1V)
ADLAR – Az eredmény igazítása **0:** jobbra, **1:** balra
MUX – bemenetválasztás (**0000-0111:** A0-A7, **1000:** belső hőmérő, **1110:** 1,1V-os referencia, **1111:** GND)

- **ADCH** és **ADCL** adatregiszterek

ADLAR = 0

15	14	13	12	11	10	9	8	
–	–	–	–	–	–	ADC9	ADC8	ADCH
ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	ADC1	ADC0	ADCL
7	6	5	4	3	2	1	0	

ADLAR = 1

15	14	13	12	11	10	9	8	
ADC9	ADC8	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	ADCH
ADC1	ADC0	–	–	–	–	–	–	ADCL
7	6	5	4	3	2	1	0	

Az ADC regiszterei

- **ADCSRA** – vezérlő és állapotregiszter I.

7	6	5	4	3	2	1	0	
ADEN	ADSC	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0	ADCSRA
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
0	0	0	0	0	0	0	0	

ADEN: ADC engedélyezése, **ADSC:** konverzió indítás, **ADIF:** konverzió vége jelzőbit, **ADIE:** megszakításkérés engedélyezés, **ADPS[2:0]:** előszámláló választás (/2 … /128)

- **ADCSRB** – vezérlő és állapotregiszter II.

7	6	5	4	3	2	1	0	
–	ACME	–	–	–	ADTS2	ADTS1	ADTS0	ADCSRB
R	R/W	R	R	R	R/W	R/W	R/W	
0	0	0	0	0	0	0	0	

ACME – az analóg komparátorhoz rendeli a bemeneti multiplexert (ADEN = 0 esetén)
ADTS[2:0] – konverziót indító (trigger) jelforrás választása:

000: Szabadonfutó mód (ADIF)	100: Timer0 Túlcsordulás
001: Analóg komparátor	101: Timer1 Compare esemény B
010: Külső megszakítás (INT0)	110: Timer1 Túlcsordulás
011: Timer0 Compare esemény A	111: Timer1 Capture esemény

Az ADC regiszterei

DIDR0 – Digital Input Disable Register 0

Bit	7	6	5	4	3	2	1	0	
(0x7E)	–	–	ADC5D	ADC4D	ADC3D	ADC2D	ADC1D	ADC0D	DIDR0
Read/Write	R	R	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- Ha 1-et írunk valamelyik bitbe, az letiltja a hozzá tartozó **A0 – A5** kivezetés digitális bemeneti bufferét. **A6** és **A7** csak analóg bemenet, ezeknél nincs mit letiltani…

A beépített hőmérő használata

- **ATmega328p** beépített hőmérő a 0b1000 ADC csatornán érhető el

```
long readTemp() {
  long result;
  // Hőmérés , 1V1 belső referenciával
  ADMUX = _BV(REFS1) | _BV(REFS0) | _BV(MUX3);
  delay(20);                            // Vref beállási idő
  ADCSRA |= _BV(ADSC);                  // Konverzió indítása
  while (bit_is_set(ADCSRA,ADSC));      // Konverzió végére vár
  result = ADCL;                        // Előbb ezt kell kiolvasni
  result |= ADCH<<8;                    // Magasabb helyiértékű bitek
  result = (result - 125) * 1075;       // Hőmérséklet * 10 000 Celsius fokban
  return result;
}

void setup() {
  Serial.begin(9600);
  Serial.println("Belső hőmérő");
}

void loop() {
  long temp = readTemp();
  Serial.print("Temperature = ");
  Serial.print(temp / 10000.0f,1);
  Serial.println(" °C");
  delay(5000);
}
```

Mérés belső referenciával, a 0b1000 csatornában

beepitett_homero.ino

Eredmények
(egy kis melegítéssel)

Belső hőmérő
Temperature = 25.6 °C
Temperature = 25.6 °C
Temperature = 25.8 °C
Temperature = 25.8 °C
Temperature = 25.9 °C
Temperature = 26.1 °C
Temperature = 25.9 °C
Temperature = 25.7 °C
Temperature = 25.8 °C
Temperature = 26.8 °C
Temperature = 26.9 °C
Temperature = 27.5 °C

A tápfeszültség meghatározása

- **ATmega328p** belső referenciája a 0b1110 ADC csatornán érhető el

tapfeszultseg_meghatarozasa.ino

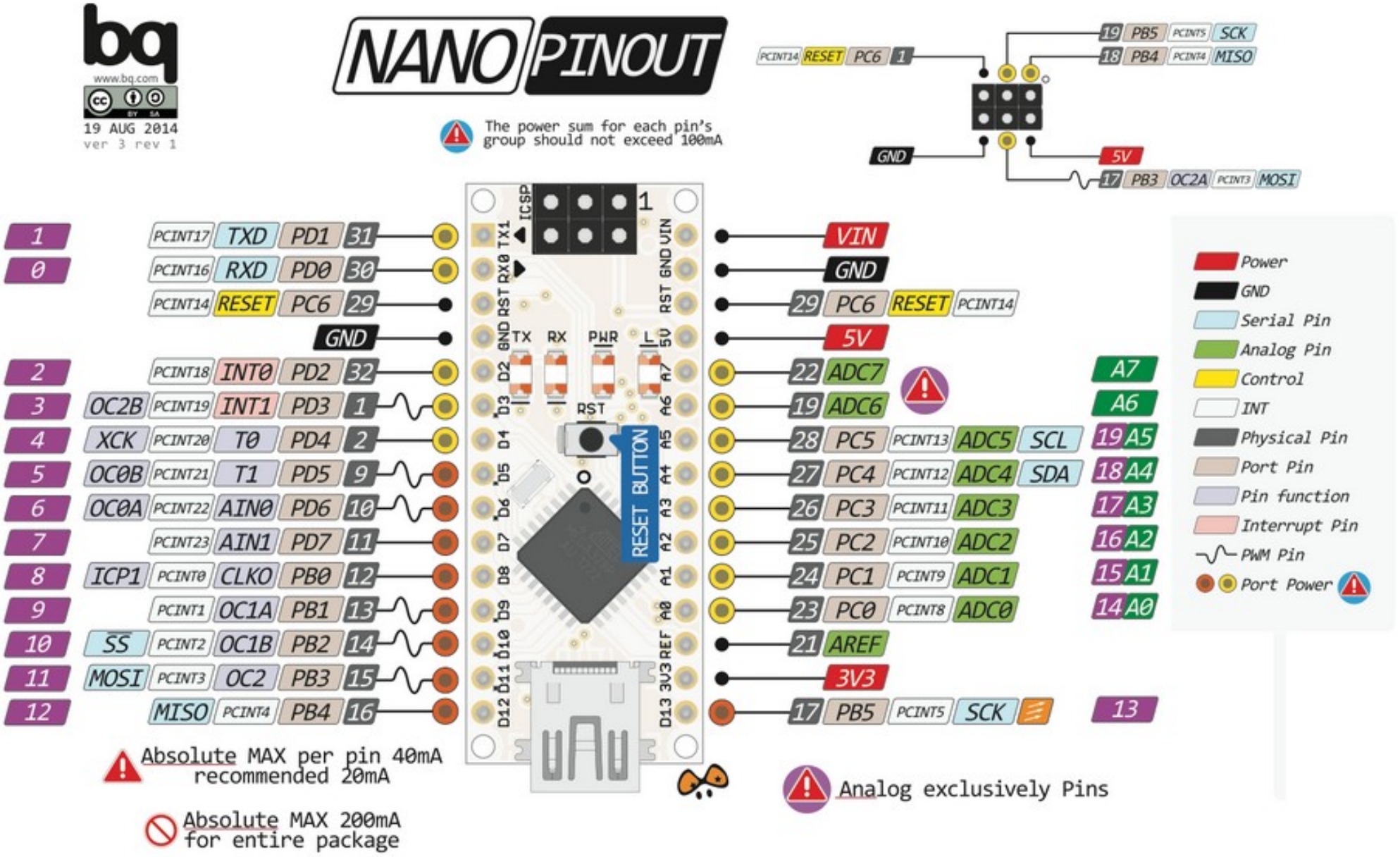
```
long readVcc() {
  long result;
  // 1.1V referenciafeszültség mérése, AVcc a referencia
  ADMUX = _BV(REFS0) | _BV(MUX3) | _BV(MUX2) | _BV(MUX1);
  delay(20);                            // Vref beállási idő
  ADCSRA |= _BV(ADSC);                  // Konverzió indítása
  while (bit_is_set(ADCSRA,ADSC));      // Konverzió végére vár
  result = ADCL;                        // Előbb ezt kell kiolvasni
  result |= ADCH<<8;                    // Magasabb helyiértékű bitek
  result = 1100L * 1024L / result;      // AVcc kiszámítása [mV]
  return result;                        // 1100 mV Vref * 1024 ADC felbontás
}
void setup() {
  Serial.begin(9600);
  Serial.println("Tápfeszültség meghatározása belső referenciával");
}

void loop() {
  long vcc = readVcc();
  Serial.print("Vcc = ");
  Serial.print(vcc);
  Serial.println(" mV");
  delay(5000);
}
```

Tápfeszültség meghatározása belső referenciával
Vcc = 4752 mV
Vcc = 4752 mV
Vcc = 4752 mV
Vcc = 4752 mV
Vcc = 4752 mV
Vcc = 4752 mV

Futási eredmény

Az Arduino nano kártya kivezetései



Ellenállás színkódok