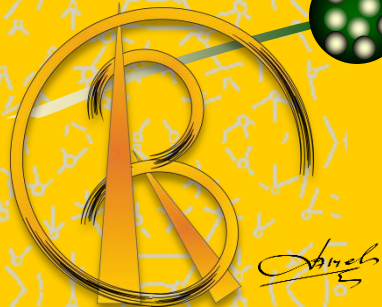
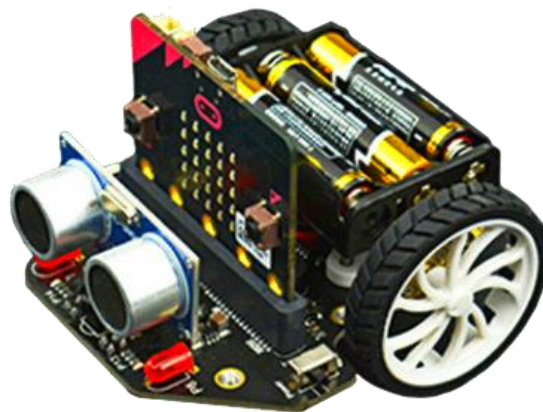
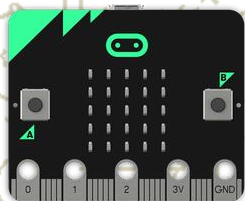


awesome  
micro:bit

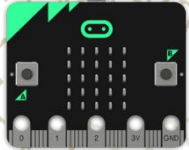


# BBC:MicroBit. Основни елементи



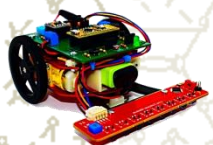


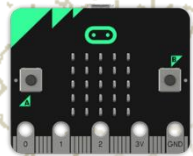
# 1. PWM сигнал



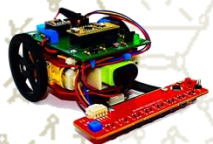
Цифровите сигнали са сигнали, които могат да бъдат представени с 0 или 1. Аналоговите сигнали от друга страна имат по-голям диапазон от възможни стойности, отколкото просто 0 или 1. И двата сигнала се използват в електрониката около нас, но се обработват много различно. Ако трябва да вземем аналогов вход, можем да получим аналоговите данни в реално време от сензор и след това с помощта на **аналогово-цифров преобразувател (ADC)** да ги преобразуваме в цифрови данни за микроконтролер;

SOCIETY  
ROBOTIC-





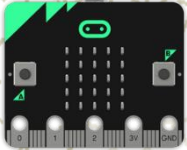
SOCIETY  
ROBOTIC—



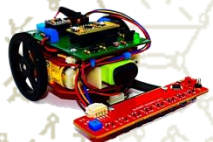
Но какво ще стане, ако трябва да управляваме аналогово устройство от нашия микроконтролер? Някои микроконтролери имат вграден **цифрово-аналогов преобразувател (DAC)** за извеждане на истински аналогов сигнал, за да управляват аналогови устройства и дори можем да използваме външен DAC. Но DAC е сравнително скъп за производство по отношение на разходите и също така заема много силициева площ. За да преодолеем тези проблеми и лесно да постигнем функционалността на DAC по много по-рентабилен начин, можем да използваме техниката на PWM.



Society of Robotics



SOCIETY OF ROBOTICS



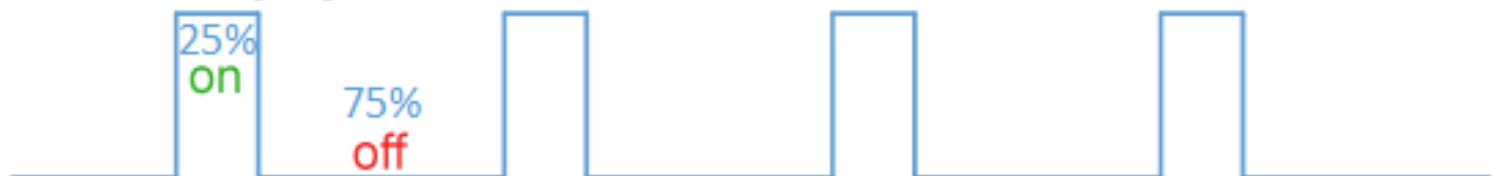
**50% duty cycle**

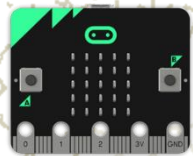


**75% duty cycle**

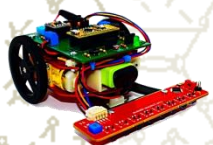


**25% duty cycle**





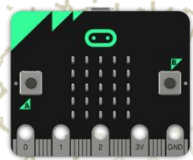
SOCIETY  
ROBOTICS



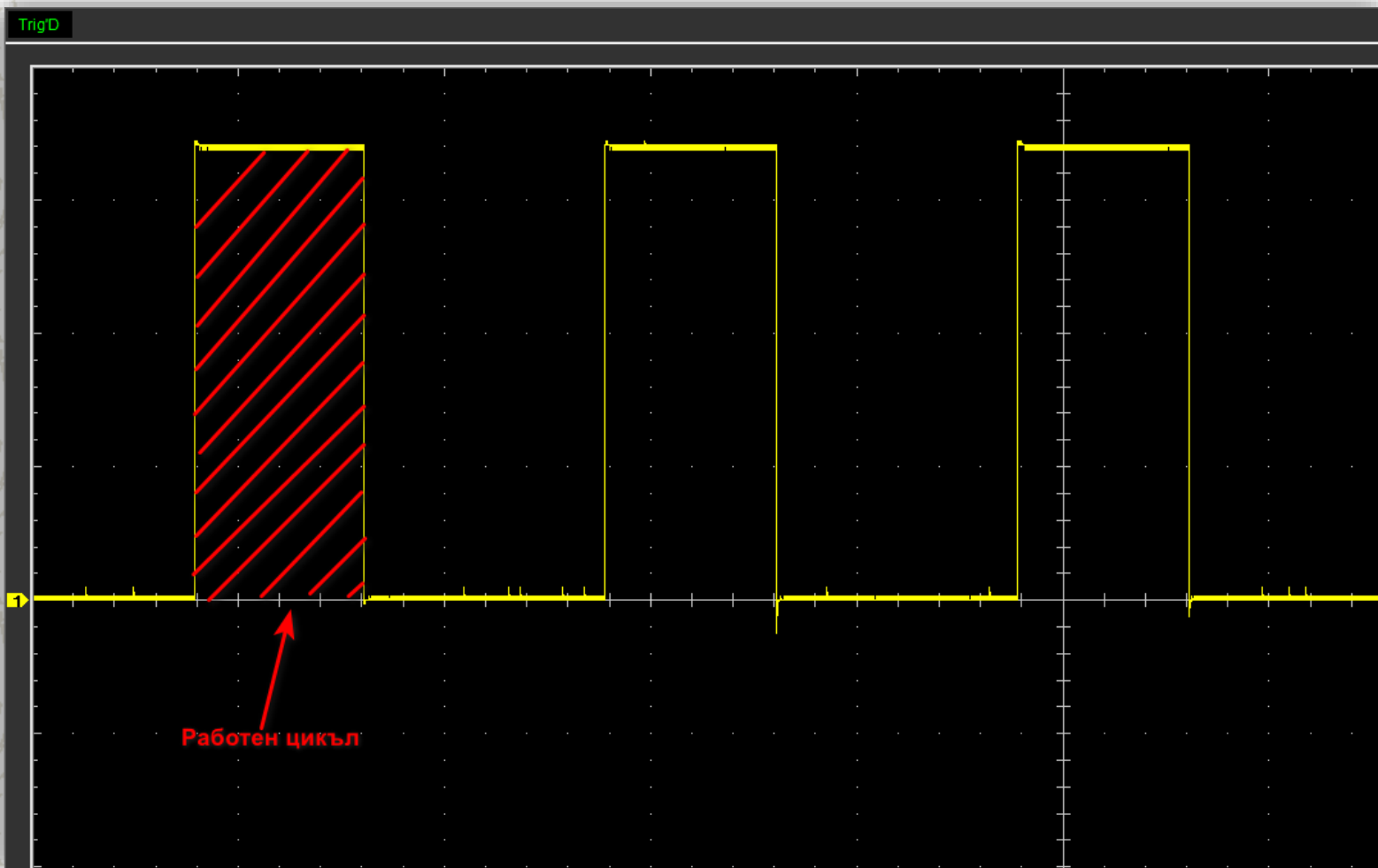
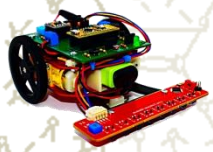
- Широчинно-импулсната модулация (PWM) използва правоъгълна импулсна вълна , чиято ширина на импулса е модулирана, което води до промяна на средната стойност на формата на вълната;
- Терминът **работен цикъл** описва съотношението на „**включено**“ време спрямо редовния интервал или „период“ от време и се измерва в **проценти**. Нисък процент работен цикъл съответства на ниска мощност, тъй като захранването е изключено през по-голямата част от времето;



Abriel

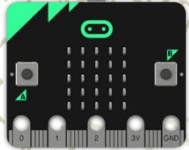


SOCIETY  
ROBOTIC

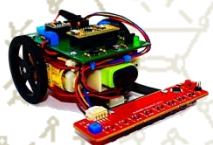
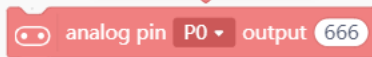
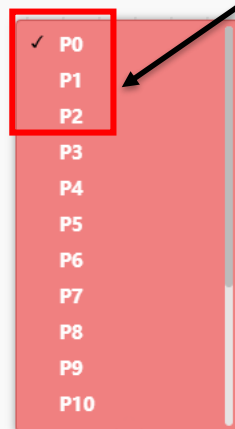




# 1.1.Блок за управление



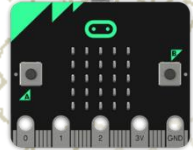
Когато използваме блока за управление трябва да изберем съответния пин и да зададем PWM стойността (0-1023);



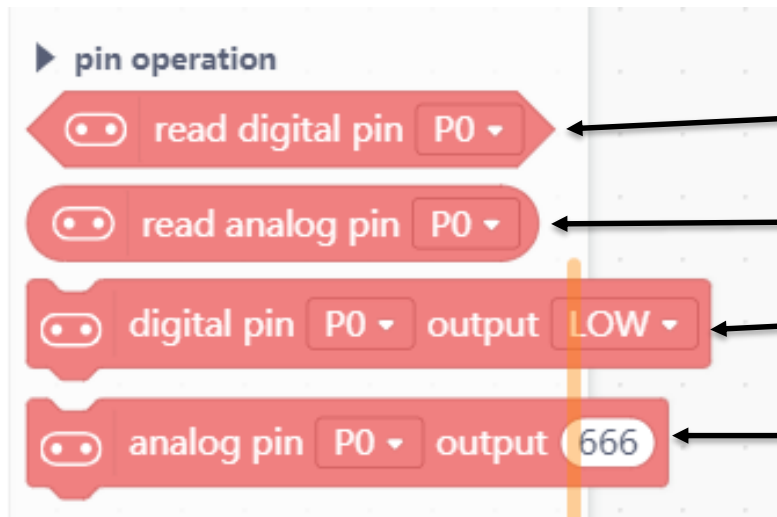




## 2.Управление на пин



- ⚡ Можем да променяме състоянието на цифров пин от **HIGH** в **LOW** и обратно;
- ⚡ Можем да прочитаме състоянието на цифров или аналогов пин;

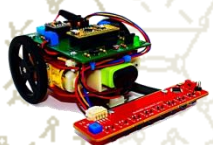


Прочитане на състоянието на цифров пин

Прочитане на състоянието на аналогов пин

Промяна на състоянието на цифров пин

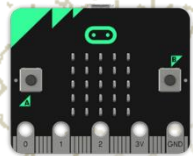
Задаване на PWM на аналогов пин







## 2.1.Функция MAP



- Функцията **map()** пренасочва число от един диапазон към друг;
- Не ограничава стойностите в диапазона, тъй като стойностите извън диапазона понякога са предвидени и полезни;

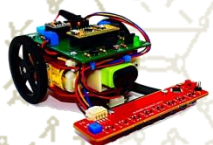


променлива

от интервал

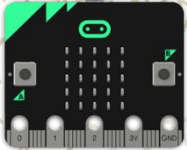
в интервал

SOCIETY  
ROBOTIC-

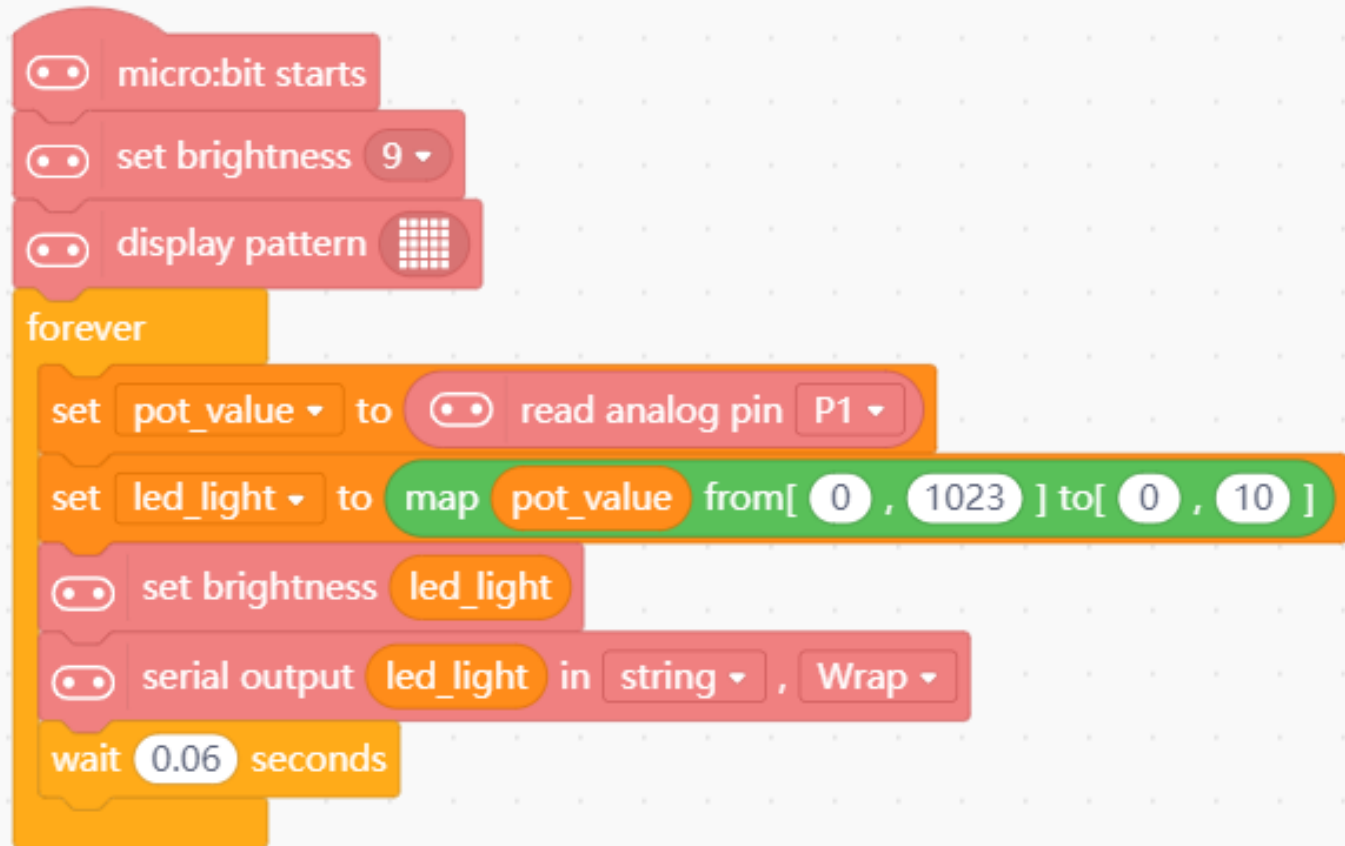
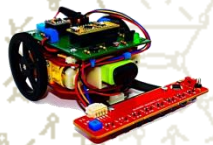




Society of Robotics

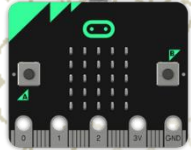


SOCIETY  
ROBOTIC





## 2.2.Функция Constrain



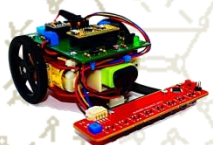
Функцията **constrain()** се използва, за да се ограничи дадена стойност в определен диапазон;



стойност

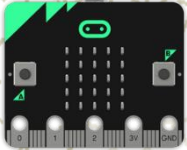
Долна граница  
на диапазона

Горна граница  
на диапазона

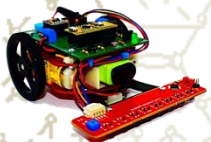




Abirah



SOCIETY  
ROBOTIC



micro:bit starts

forever

set pot\_value to read analog pin P1

set old\_value to map pot\_value from [ 0 , 1023 ] to [ 0 , 500 ]

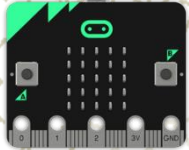
set new\_value to constrain old\_value between (min) 0 and (max) 255

serial output new\_value in string , Wrap

wait 0.5 seconds

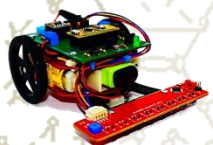


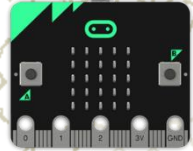
## 3.3 зумер



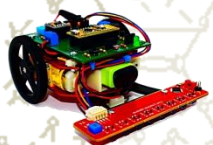
Пиезоелектричните зумери се състоят от пиезоелектрична диафрагма, която представлява пиезоелектричен керамичен диск прилепен към метална плоча от месинг или никелова сплав. Около диска има резонираща кутия с отвор в средата за емитиране на звука. В задната част на елемента има два извода. Единият от тях е свързан с металният диск, а другият с пиезопластината;

SOCIETY  
ROBOTICS

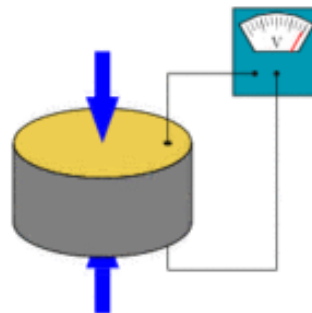




SOCIETY  
ROBOTICS

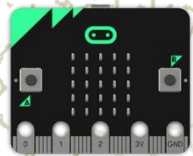


При подаване на напрежение пиезокристалът се разширява и свива увеличавайки със себе си металния диск, който произвежда звук генериран от вибрациите в него. Звукът се усилва от акустичната кутия, в която се намира и излиза през отвора в средата ѝ;

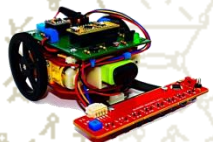


## 3.1. Възпроизвеждане на мелодия

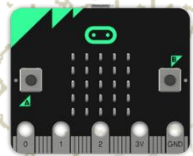
- ⚡ За да предадем **честотата на звука (Hz)**, която цифровият сигнал се опитва да възпроизведе, **ние променяме времето между високото или ниското напрежение на сигнала;**
- ⚡ За да направите това, трябва да се намери **периодът (ms)** на нотата, което е времето, необходимо на вълната да премине веднъж. Това се прави, като се вземе обратната стойност на честотата на нотата;



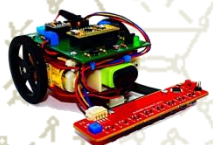
SOCIETY  
ROBOTIC-







SOCIETY  
ROBOTIC-



Период (ms)=1/Честота (Hz)

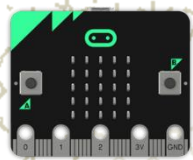
- Например, нотата A4 се възпроизвежда на 440 Hz, така че използвайки нашето изчисление:

$$1/440 = 2,273\text{ms}$$

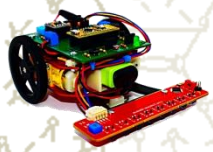
- След като имаме стойността на периода, за да представим височината в PWM, ние просто държим сигнала висок за половината от дължината на периода (1,13636ms) и нисък за другата половина (1,13636ms).



Abriel



SOCIETY  
ROBOTIC



Задаваме пин за зумера и възпроизвеждане на мелодия

Задаваме пин за зумера и възпроизвеждане на мелодия

Задаваме пин за зумера и възпроизвеждане на нота

Промяна на темпото с определена стойност

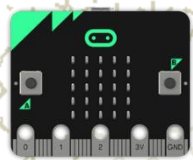
Задаване на стойност на темпото

Спиране на мелодията

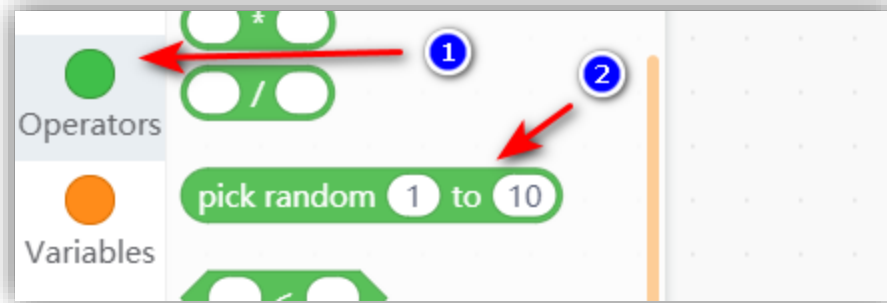
Получаване на стойността на темпото



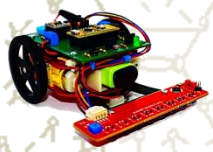
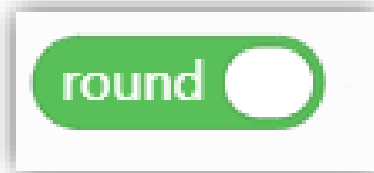
## 4.Функции

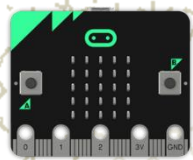


- Функцията **random()** генерира псевдослучайно число в определен интервал;

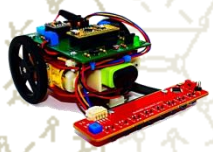


- Функцията **round()** закръгля дадено число;

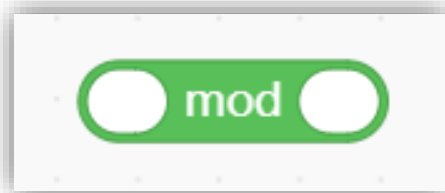




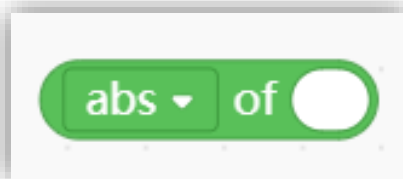
SOCIETY  
ROBOTICS



Функцията **mod()** извършва деление с остатък между две числа;

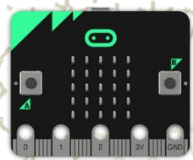


Функцията **abs()** връща като резултат абсолютната стойност на числото, зададено като параметър на функцията;

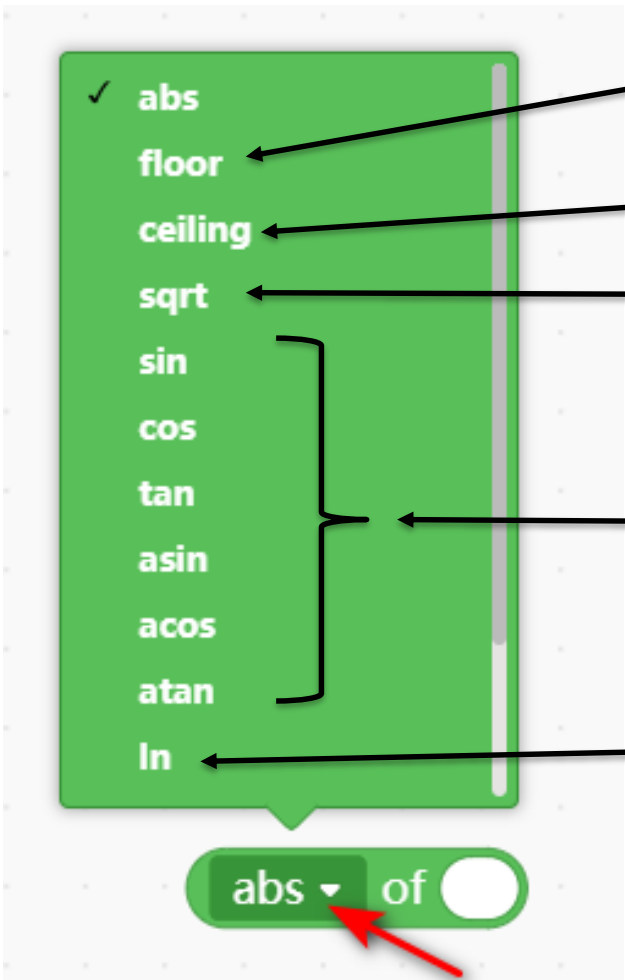
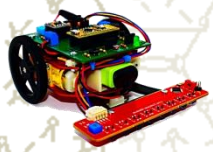




Abriel



SOCIETY  
ROBOTIC



Закръгля число до най-  
близкото по-малко цяло число

Закръгля число до най-  
близкото по-голямо цяло число

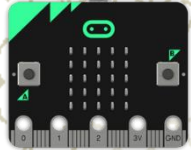
Корен квадратен

Тригонометрични функции

Логаритъм



## 4.1.Преобразуване на данни



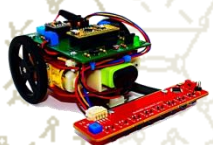
- Можем да преобразуваме число в текст със следната функция:

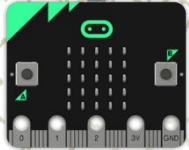
convert number 132 to string

- Можем да преобразуваме текст в цяло или реално число със следната функция:

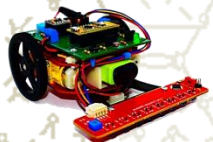
convert string "123" to Integer ▾

✓ Integer  
Decimal





**SOCIETY  
ROBOTIC**



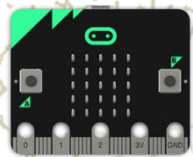
# 5.Акселерометър

Weightless  
State

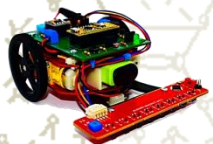


$X=0g$   
 $Y=0g$   
 $Z=0g$





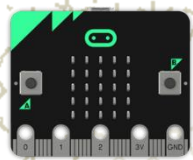
SOCIETY  
ROBOTIC-



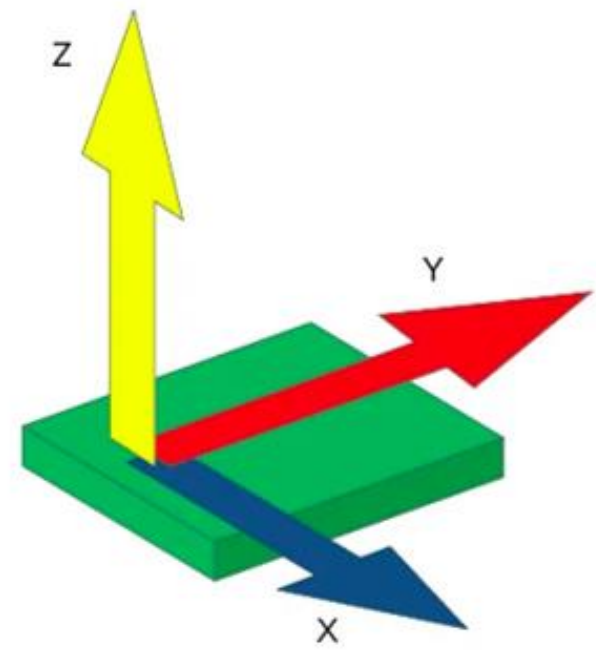
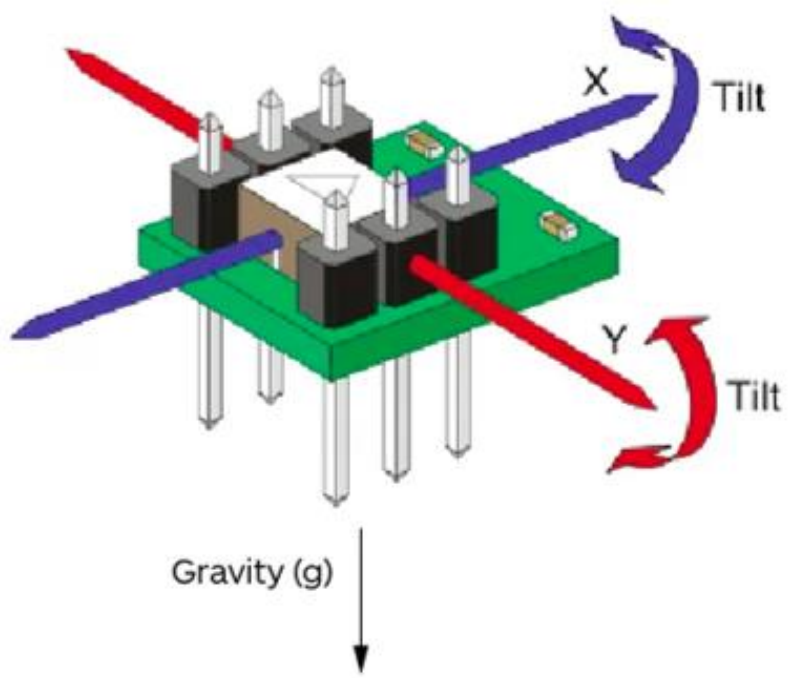
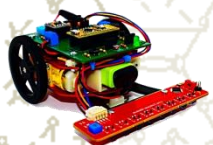
- 💡 Да предположим, че кубът е в космоса, където всичко е в безтегловно състояние, топката просто ще се носи в средата на куба.
- 💡 Сега нека си представим, че всяка стена представлява определена ос.
- 💡 Ако внезапно преместим кутията наляво с ускорение  $1g$  (една G-сила  $1g$  е еквивалентна на гравитационно ускорение  $9,8 \text{ m/s}^2$ ), без съмнение топката ще удари стената X. Ако измерим силата, която топката прилага стената X, можем да получим изходна стойност от  $1g$  по оста X.



Society of Robotics



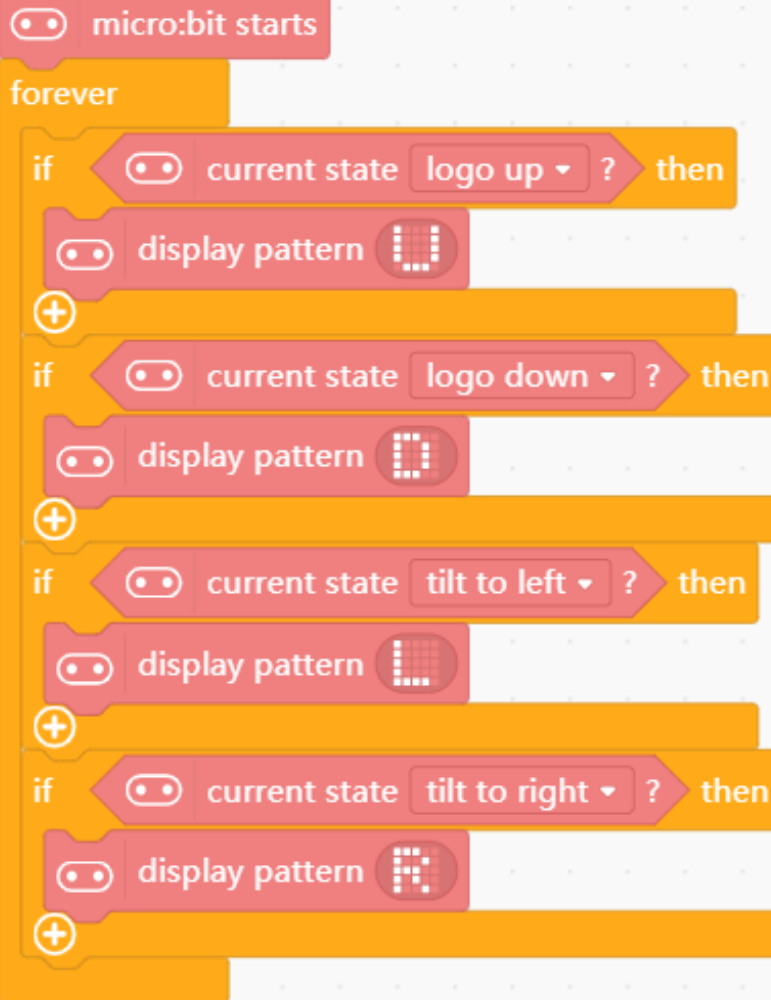
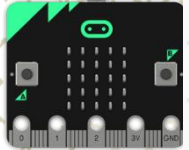
SOCIETY OF ROBOTICS





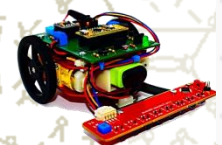
# Ориентация лого

orient\_logo\_10



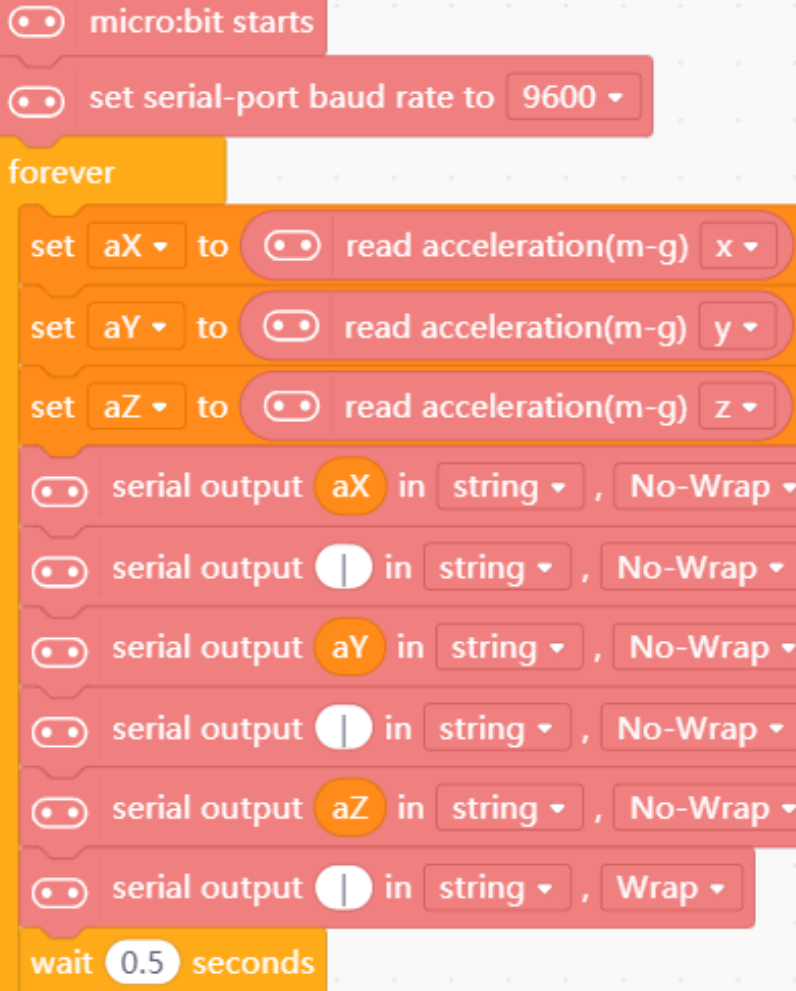
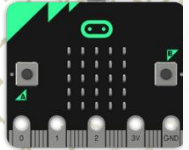
```
6 #include <Microbit_Matrix.h>
7 #include <Microbit_Sensors.h>
8 // Static constants
9 const uint8_t bbcBitmap[][5] = {
10     {B10001,B10001,B10001,B10001,B01110},
11     {B11100,B10010,B10010,B10010,B11100},
12     {B10000,B10000,B10000,B10000,B11110},
13     {B11100,B10010,B11100,B10100,B10010}
14 };
15
16 // Main program start
17 void setup() {
18 }
19 void loop() {
20     if ((Sensors.getGesture(Sensors.LogoUp))) {
21         MMatrix.show(bbcBitmap[0]);
22     }
23     if ((Sensors.getGesture(Sensors.LogoDown))) {
24         MMatrix.show(bbcBitmap[1]);
25     }
26     if ((Sensors.getGesture(Sensors.TiltLeft))) {
27         MMatrix.show(bbcBitmap[2]);
28     }
29     if ((Sensors.getGesture(Sensors.TiltRight))) {
30         MMatrix.show(bbcBitmap[3]);
31     }
32 }
33
34 }
```

SOCIETY  
ROBOTIC

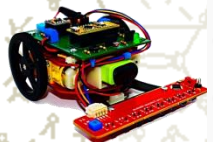




# Ускорения по осите

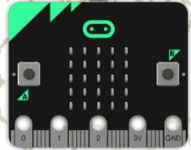


```
6 #include <Microbit_Sensors.h>
7
8 // Dynamic variables
9 volatile float mind_n_aX, mind_n_aY, mind_n_aZ;
10
11
12 // Main program start
13 void setup() {
14     Serial.begin(9600);
15 }
16 void loop() {
17     mind_n_aX = (Sensors.acceleration(Sensors.X));
18     mind_n_aY = (Sensors.acceleration(Sensors.Y));
19     mind_n_aZ = (Sensors.acceleration(Sensors.Z));
20     Serial.print(mind_n_aX);
21     Serial.print(" | ");
22     Serial.print(mind_n_aY);
23     Serial.print(" | ");
24     Serial.print(mind_n_aZ);
25     Serial.println(" | ");
26     delay(500);
27 }
```





## 6. Цифров компас

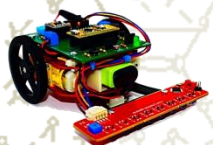


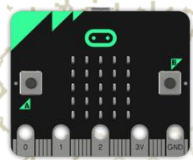
☀ Магнитният компас е магнитен навигационен уред за ориентиране в местност, чрез определяне на посоките на света;

☀ Показания

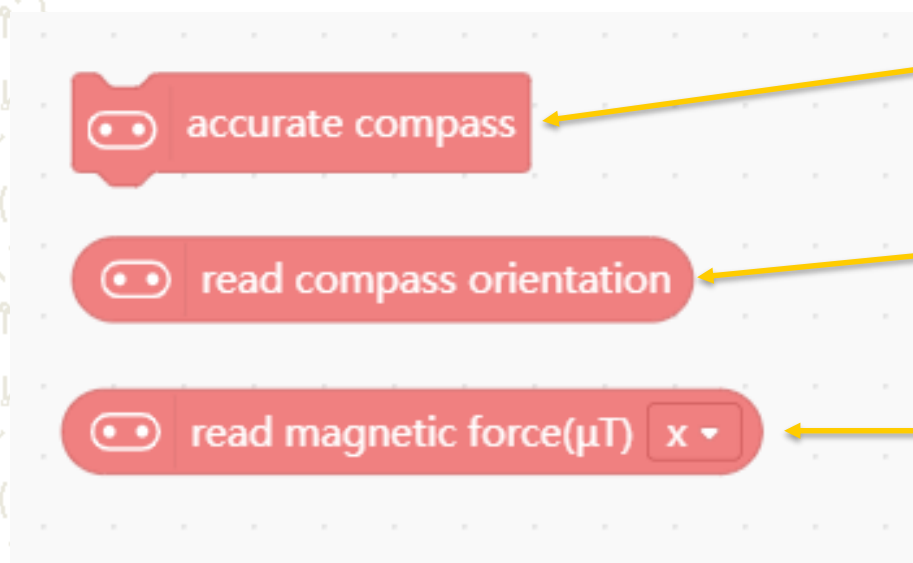
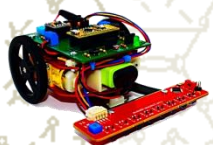
0 - 22	North (север)
23 - 68	NE (североизток)
69 - 113	East (изток)
114 - 158	SE (югоизток)
159 - 201	South (юг)
202 - 248	SW (югозапад)
249 - 291	West (запад)
292 - 338	NW (северозапад)

SOCIETY  
ROBOTIC





SOCIETY  
ROBOTIC



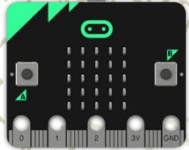
Калибриране на компаса

Прочитане на показанията  
от компаса

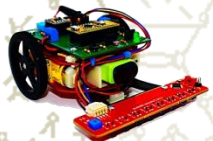
Прочитане на показанията  
на магнитната сила от компаса



Abir



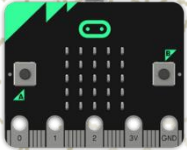
SOCIETY  
ROBOTIC



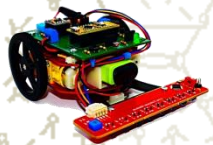
```
micro:bit starts
accurate compass
forever
  serial output read compass orientation in string , Wrap
  wait 0.1 seconds
```







**SOCIETY  
ROBOTIC-**



micro:bit starts

accurate compass

forever

set compass to read compass orientation

if compass > 0 and compass < 23 then

display pattern

else if compass > 69 and compass < 114 then

display pattern

else if compass > 159 and compass < 202 then

display pattern

else if compass > 249 and compass < 292 then

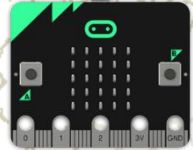
display pattern

serial output compass in string , Wrap

wait 0.1 seconds

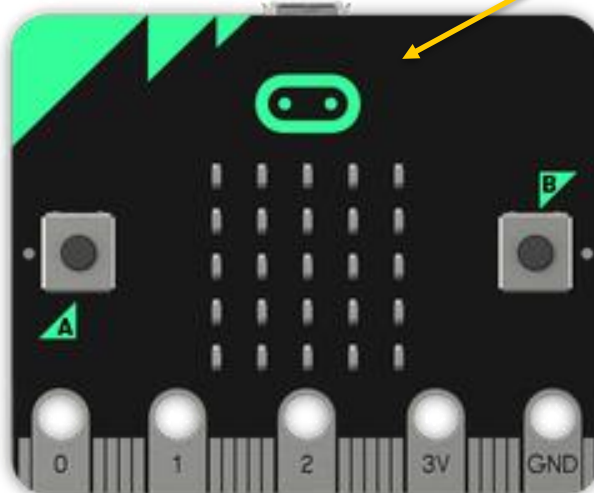


## 7.Микрофон

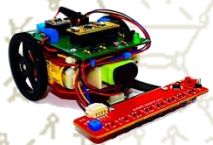


Microbit разполага с вграден микрофон за приемане на звукови сигнали. Той е свързан със специално обозначен пин;

микрофон

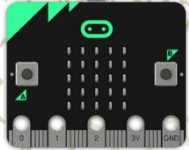


get sound level(V2)

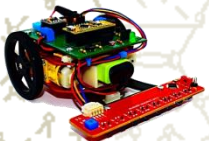




Society of Robotics



SOCIETY OF ROBOTICS



micro:bit starts

forever

set sound to get sound level(V2)

serial output Нивото на звук е: in string , No-Wrap

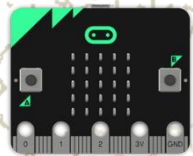
serial output sound in string , Wrap

wait 0.5 seconds

```
it-nrf5sdk\nRF5SDK\components\libraries\util -I C:\Program Fil  
C:\Program Files (x86)\Mind+\Arduino\hardware\tools\nRF5\gcc-a  
"C:\Program Files (x86)\Mind+\Arduino\hardware\tools\nRF5\gcc-  
"C:\Program Files (x86)\Mind+\Arduino\hardware\tools\nRF5\gcc-  
The project uses 91984 bytes, occupies (18%) program memory sp  
Global variables use 8896 bytes, (7%) of dynamic memory, leavi  
upload success  
Нивото на звук е: 7.00  
Нивото на звук е: 33.00  
Нивото на звук е: 26.00  
Нивото на звук е: 11.00  
Нивото на звук е: 15.00  
Нивото на звук е: 26.00
```

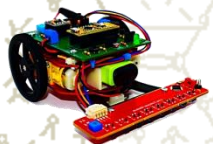


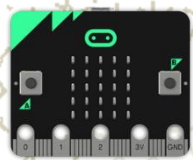
## 8.Потенциометър



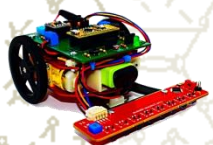
- 💡 **Резисторът**, наричан също съпротивление, е двуизводен пасивен електронен компонент, чиято основна характеристика е **електрическото съпротивление**;
- 💡 Единицата за измерване на съпротивлението на резисторите е ом, наречена в чест на немския физик Георг Ом;
- 💡 Използването на резистор в електрическа верига е с цел промяна на съпротивлението на ел. ток и от там промяна на напрежението;

SOCIETY  
ROBOTIC





SOCIETY  
ROBOTICS



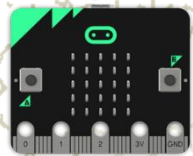
Законът на Ом е физичен закон, определящ зависимостта между напрежението, тока и съпротивлението на проводника в електрическа верига. Наречен е в чест на неговия откривател Георг Ом;

$$U = I * R$$

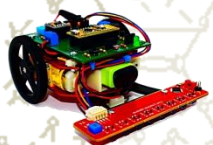
$U$  – напрежението в ел. верига

$I$  – тока в ел. верига

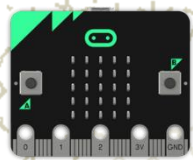
$R$  – съпротивлението в ел. верига



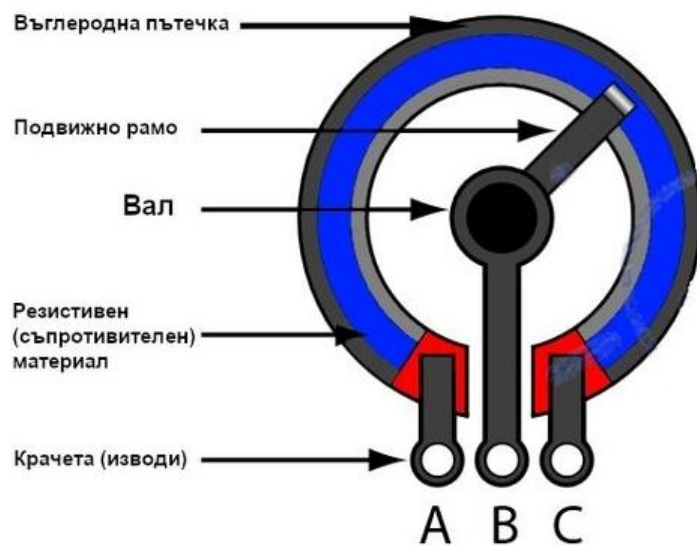
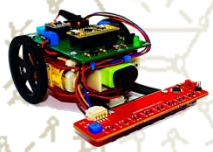
SOCIETY  
ROBOTICS



- 💡 **Потенциометърът** е резистор с 3 извода, с който е възможно при промяната на съпротивлението чрез плъзгащ контакт в електрическата верига, да се променя изходното електрическо напрежение в предварително конструктивно зададени граници;
- 💡 Плъзгащият контакт на този пасивен компонент е единия от изходните електроди и работи като **делител на напрежение**;



**SOCIETY  
ROBOTIC**



Фиг. 1

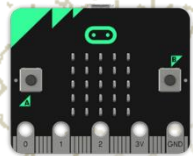


## Конструкция на потенциометър

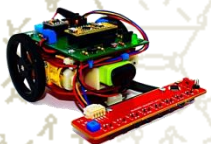


## 9. Сензор за температура DS18B20

- Цифровият термометър **DS18B20** осигурява от 9-битови до 12-битови измервания на температурата по Целзий;
- DS18B20 комуникира по **1-Wire шина**, която по дефиниция изисква само една линия за данни (и заземяване) за комуникация с централен микропроцесор;
- Всеки DS18B20 има уникален **64-битов сериен код**, който позволява на множество DS18B20 да функционират на една и съща 1-Wire шина;

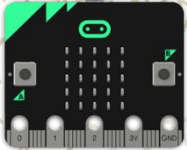


SOCIETY  
ROBOTIC

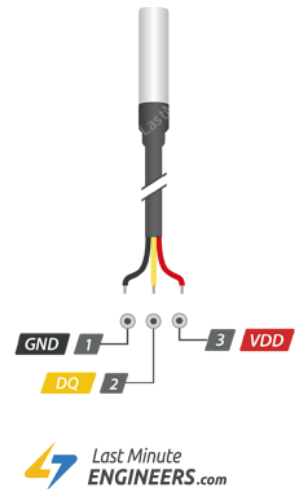
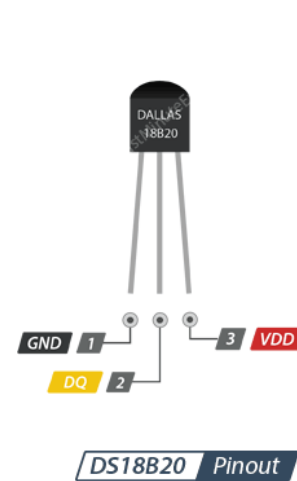
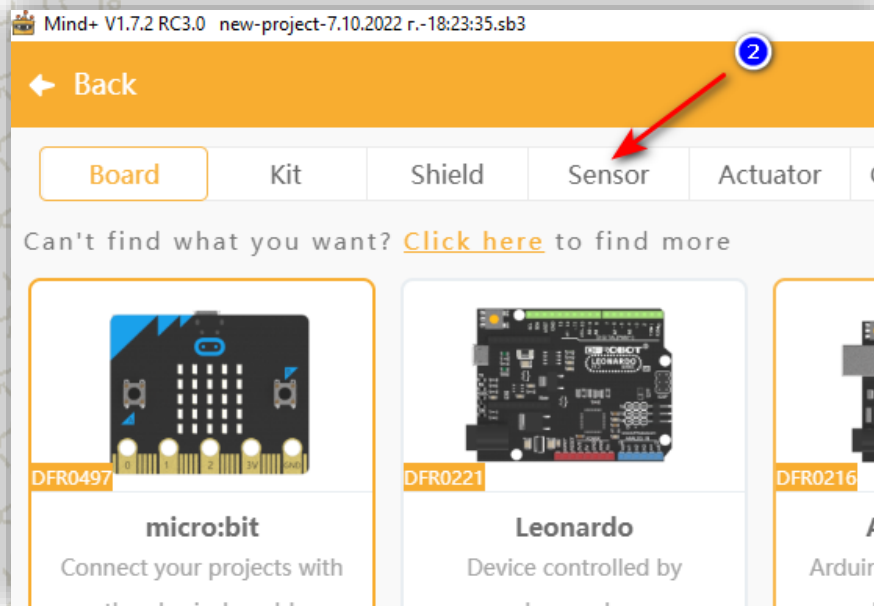
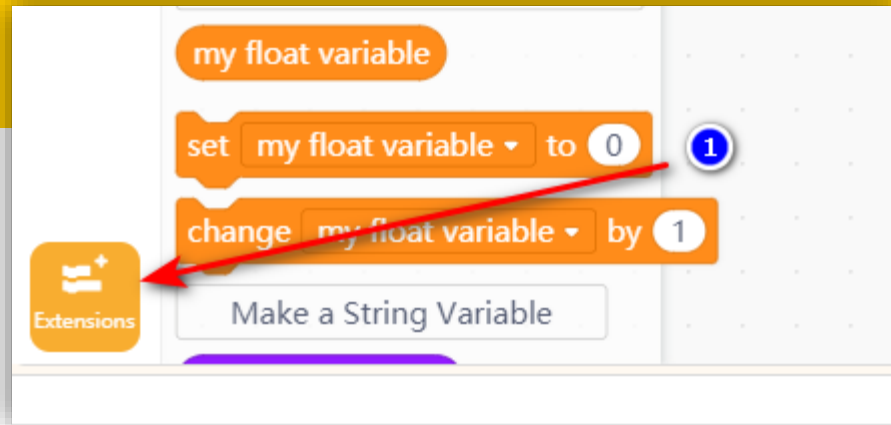
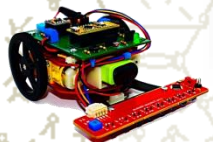




Society of Robotics

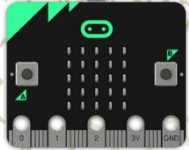


SOCIETY OF ROBOTICS

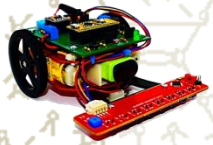






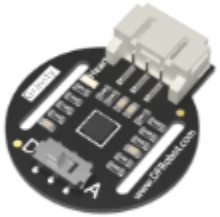
Society of Robotics



SOCIETY OF ROBOTICS

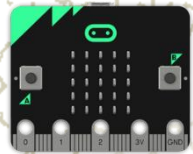


### Select Sensor

Communication	Display	Function	Internet	User-Ext
<div>3</div>				
 <p>0067 SEN0137</p> <p><b>HT11/22 Temperature and Humidity</b></p> <p>Detect environment temperature and humidity</p>	 <p>DFR0024 KIT0021 DFR0198</p> <p><b>DS18B20 Temperature Sensor</b></p> <p>Detect ambient temperature with large range of -55~+125°C</p>	 <p>SEN0203</p> <p><b>Heart Rate Monitor Sensor</b></p> <p>Mini heart rate sensor with analog pulse and digital square wave output mode</p>		

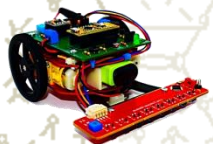


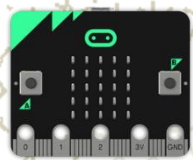
## 10.WiFi комуникация



- ⚡ Централният процесор (CPU) на microbit v2 е **Nordic Semiconductor nRF52833**. Освен компютърен процесор с общо предназначение, този чип съдържа и вграден 2.4GHz радио модул.
- ⚡ Това радио може да бъде конфигурирано по различни начини и е предназначено основно да работи с **Bluetooth Low Energy (BLE)** протокол. Въпреки това, той може да бъде поставен и в много по-опростен режим на работа, който позволява проста, директна комуникация от microbit към microbit.

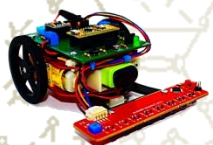
SOCIETY  
ROBOTICS

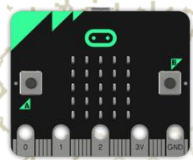




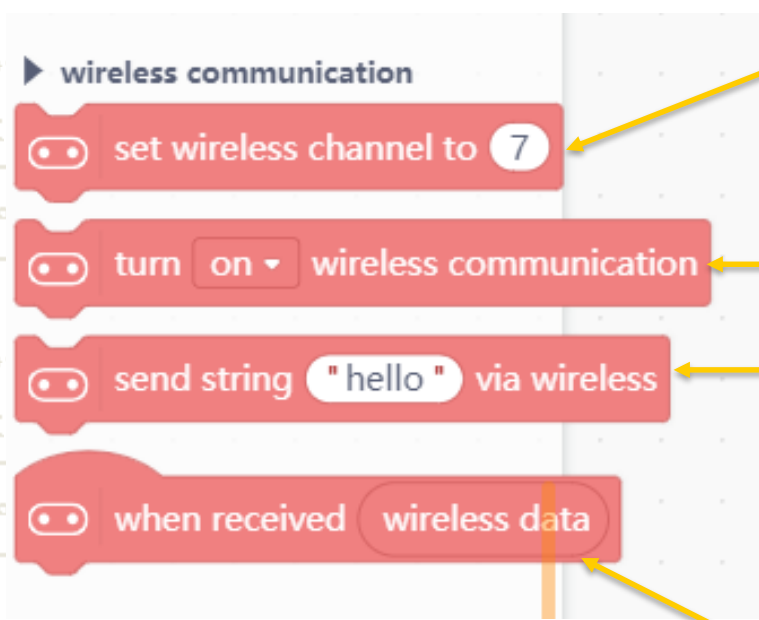
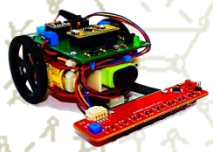
- Компонентът MicroBitRadio се състои от три класа - **MicroBitRadio**, **MicroBitRadioEvent** и **MicroBitRadioDatagram**;
- Заедно те предоставят възможност за изпращане на пакети с данни с общо предназначение от един microbit към друг и за разширяване на шина за съобщения, за да обхване множество microbits. Така ако се случи събитие на един microbit, можете да го получите на друг с помощта на нормалния механизъм за слушане.

SOCIETY  
ROBOTIC





SOCIETY  
ROBOTIC



Задава канал за комуникация  
0 - 255 канала

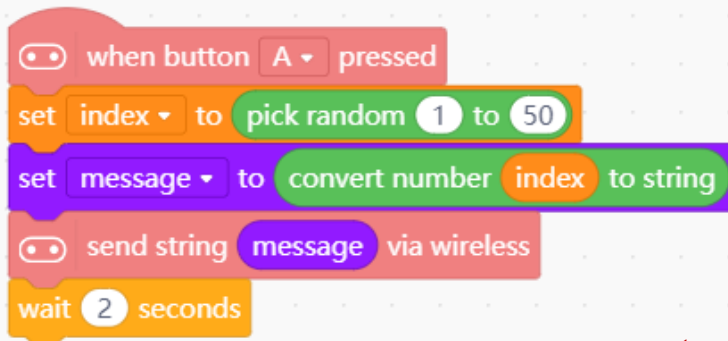
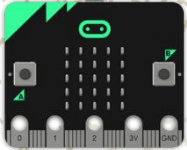
Включва/Изключва комуникация

Изпраща текстово съобщение

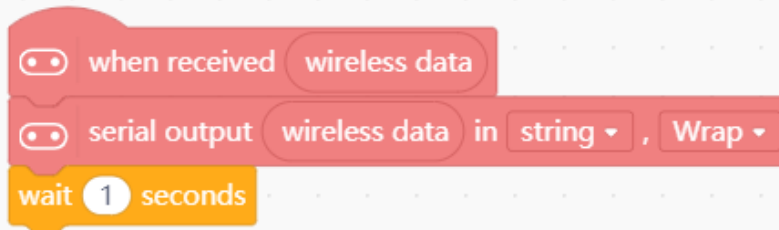
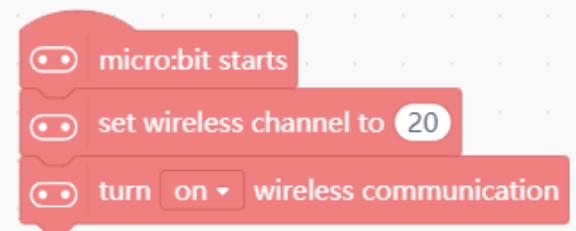
Когато получи съобщение  
го съхранява в променливата  
wireless data



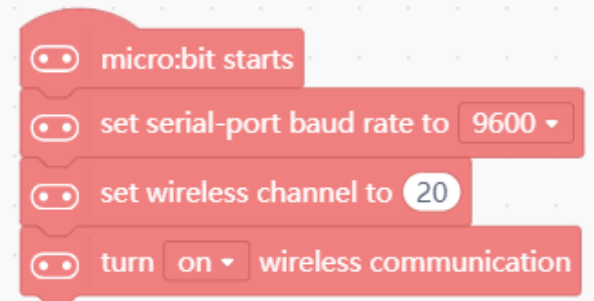
# Случайно число



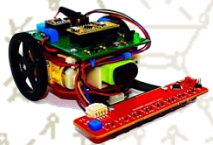
transmitter



receiver



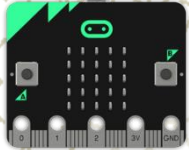
SOCIETY  
ROBOTIC







# Съобщение



```
when button A pressed
  set message to "Hello"
  send string message via wireless
  wait 2 seconds
```

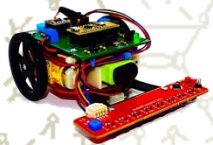
transmitter

```
micro:bit starts
  set wireless channel to 20
  turn on wireless communication
```

```
when received wireless data
  set message to wireless data
  serial output message in string , Wrap
  wait 0.1 seconds
```

receiver

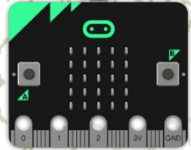
```
micro:bit starts
  set wireless channel to 20
  turn on wireless communication
```



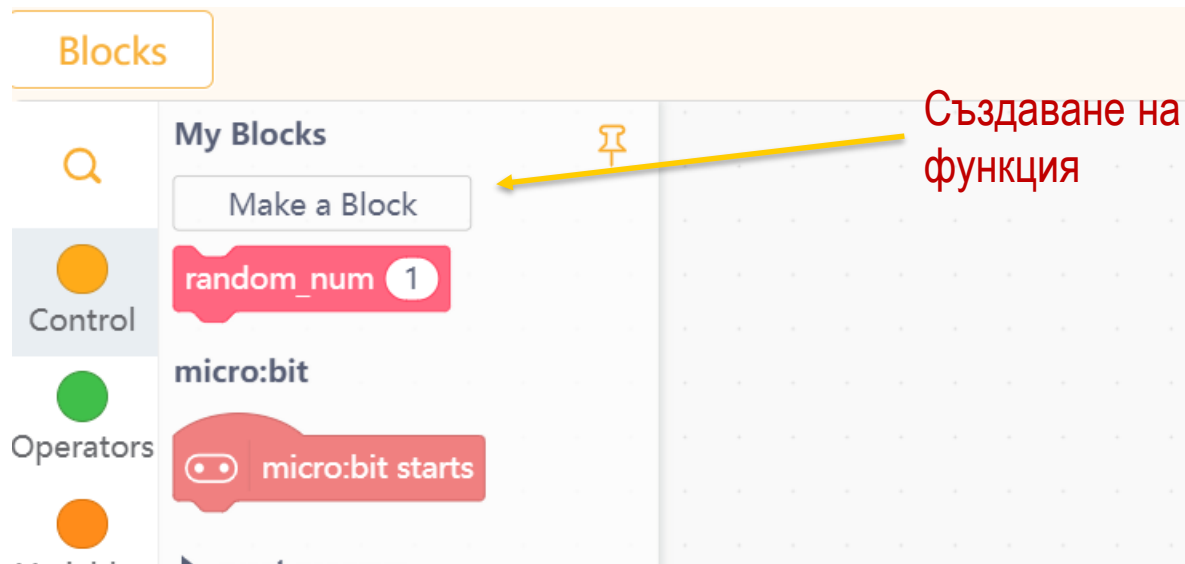




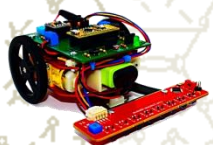
# 11. Дефиниране на функция

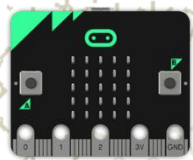


Можем да създадем собствена функция, като дефинираме съответните елементи на функцията и създадем код;

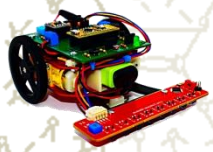


Създаване на блок за функция









SOCIETY  
ROBOTIC





### Make a Block

  
Add an input  
text

  
Add an input  
number

  
Add an input  
boolean

  
Add a label

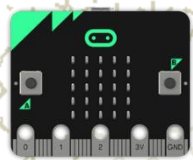
Cancel

OK

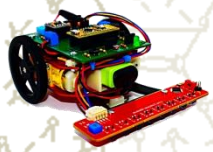
Добавяне на съответните компоненти към функцията



Abirah



SOCIETY  
ROBOTIC



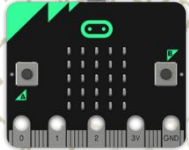
```
define random_num max_val
set rand_num to pick random 1 to max_val
serial output rand_num in string , Wrap

micro:bit starts
forever
  random_num 10
  wait 1 seconds
```

Генериране на случайно число в диапазон зададен от потребителя



## 12. RGB LEDS Neopixel

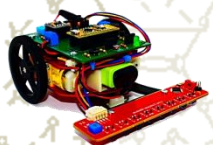


💡 **Neopixel** са модули от индивидуално адресируеми RGB светодиоди, управлявани с драйвер WS2812, работещи с напрежение от 3 до 5V;

💡 **Neopixel** позволяват последователно добавяне на светодиоди, които се управляват от едни канал (пин);



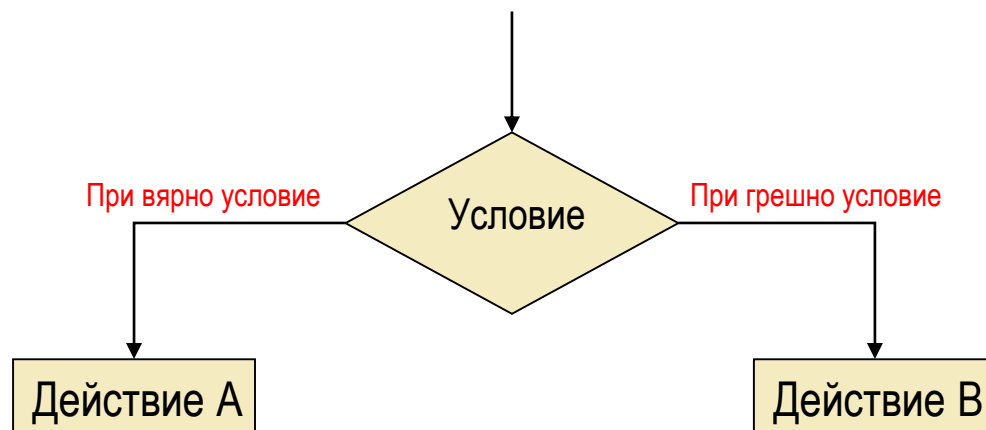
SOCIETY  
ROBOTIC

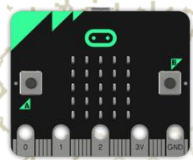


# 13. Условна структура

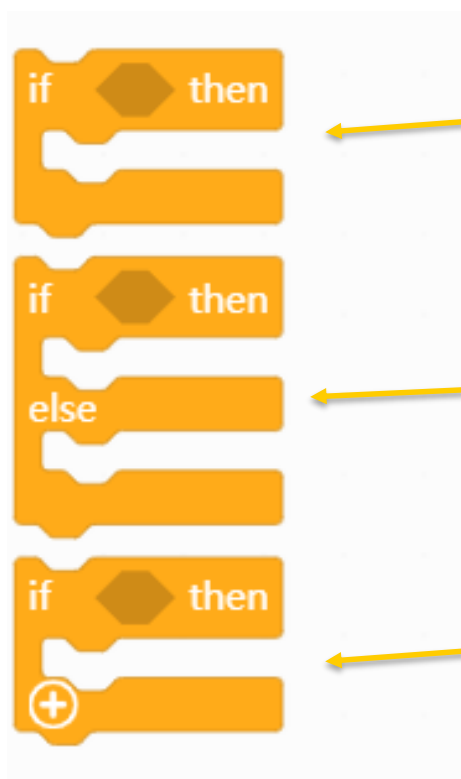
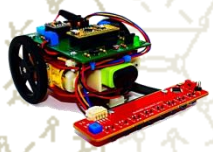
Условните изявления са известни още като изявления за вземане на решения.

Използваме тези изрази, когато искаме да изпълним блок от код, когато даденото условие е вярно или невярно;





**SOCIETY  
ROBOTIC**



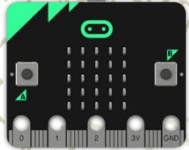
Съкратена форма на условна структура

Пълна форма на условна структура

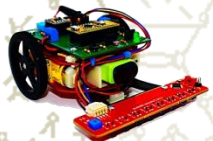
Вградени условни структури



# Оператори за сравнение



**SOCIETY  
ROBOTIC**



По-малко



По-малко или равно



Равно



По-голямо



По-голямо или равно



Логическо И



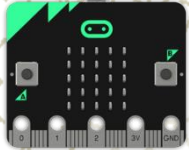
Логическо ИЛИ



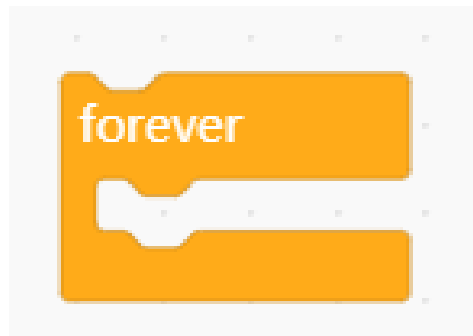
Логическо ОТРИЦАНИЕ



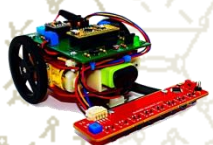
# 14.Цикъл While



- Цикълът **WHILE** не се използва за определен брой повторения, а за повторения, докато дадено условие е изпълнено;
- В **Arduino** често се използва за създаване на безкраен цикъл, като по този начин се симулира работата на микроконтролера;



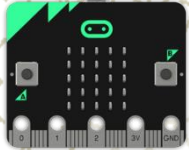
SOCIETY  
ROBOTICS



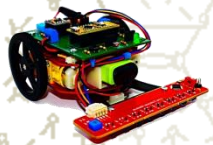




Abhishek



SOCIETY  
ROBOTIC



micro:bit starts

forever

display pattern



wait 1 seconds

display pattern

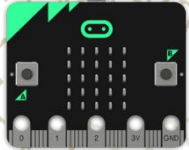


wait 1 seconds

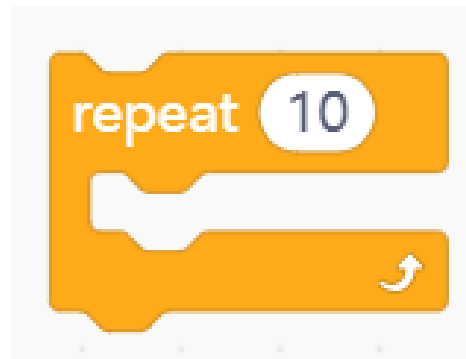
```
1 # MindPlus
2 # microbit
3 from microbit import *
4
5
6 while True:
7     display.show(Image("99999:90009:90009:90009:99999"))
8     sleep(1*1000)
9     display.show(Image("90009:09090:00900:09090:90009"))
10    sleep(1*1000)
11
```



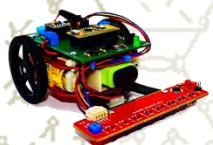
# 15.Цикъл Repeat



- Цикълът **REPEAT** (цикъл **FOR**) представлява програмна конструкция, която съдържа в себе си код, изпълнението на който се повтаря определен брой пъти;
- Цикълът **FOR** е цикъл с предварително определен брой на повторенията;

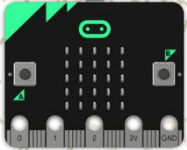


SOCIETY  
ROBOTIC

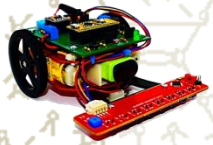




Society for Robotics



SOCIETY  
ROBOTIC



micro:bit starts

set serial-port baud rate to 9600

repeat 10

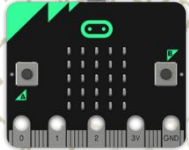
serial output hello in string , Wrap

wait 1 seconds

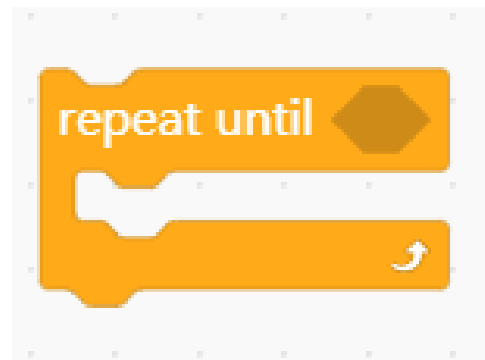
```
1 //  
2  
3  
4  
5  
6  
7 // Main program start  
8  
9 void setup() {  
10     Serial.begin(9600);  
11     for (int index = 0; index < 10; index++) {  
12         Serial.println("hello");  
13         delay(1000);  
14         yield();  
15     }  
16 }  
17 void loop() {  
18  
19 }  
20
```



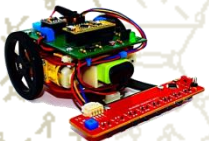
## 16.Цикъл Repeat ..Until



Цикълът **Repeat.. Until** ще се изпълнява докато условието **Е ГРЕШНО**, след което управлението на програмата ще се предаде на първия ред след края на цикъла;

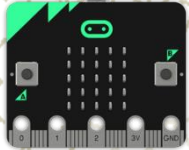


SOCIETY  
ROBOTICS

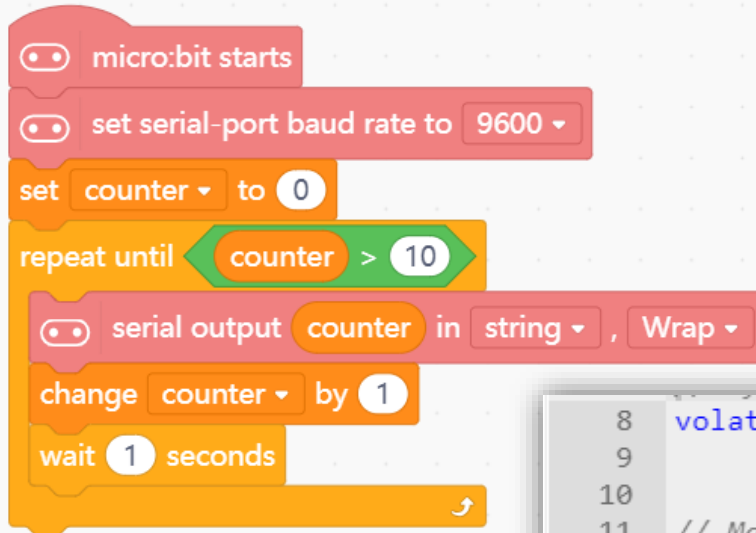
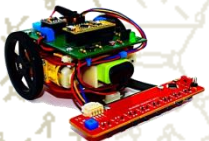




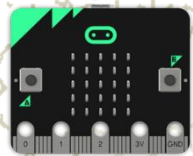
Society of Robotics



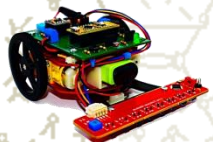
SOCIETY OF ROBOTICS



```
8 volatile float mind_n_counter;
9
10
11 // Main program start
12 void setup() {
13     Serial.begin(9600);
14     mind_n_counter = 0;
15     while (!(mind_n_counter>10)) {
16         Serial.println(mind_n_counter);
17         mind_n_counter += 1;
18         delay(1000);
19         yield();
20     }
21 }
22 void loop() {
23
24 }
```



**SOCIETY  
ROBOTIC—**



# УПРАЖНЕНИЕ