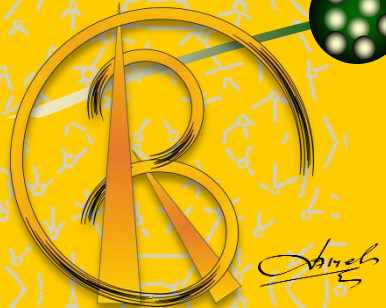
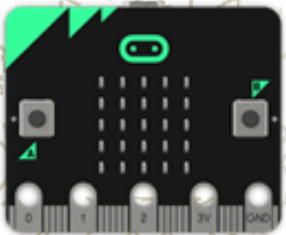


awesome  
micro:bit



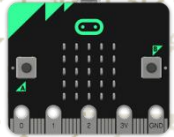
# Роботика и компютърно моделиране с MicroBit

**DFRobot – Micro IO box**

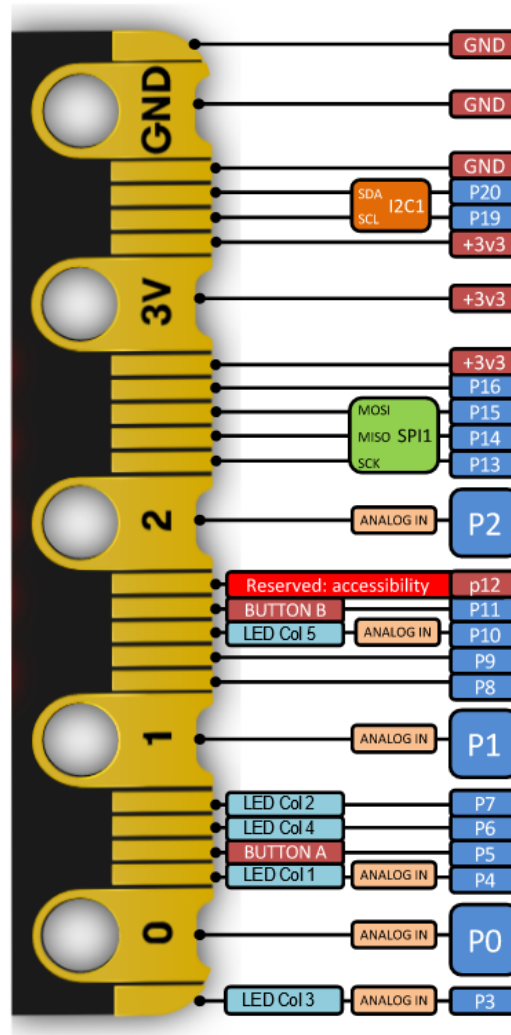
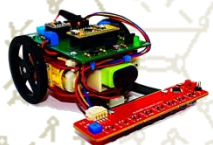




# Карта на пиновете

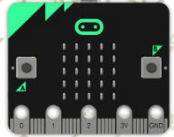


**SOCIETY  
ROBOTIC**

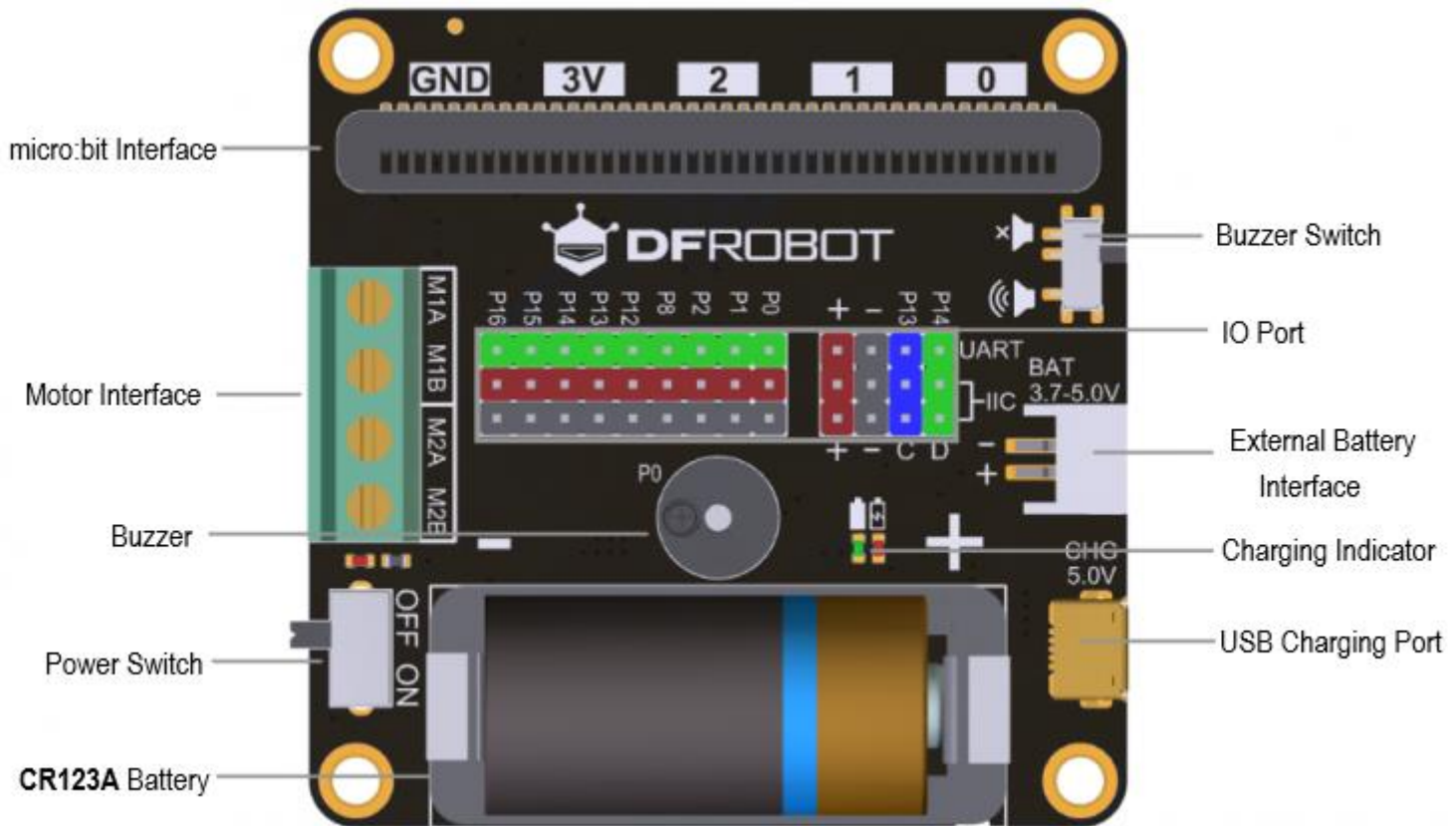
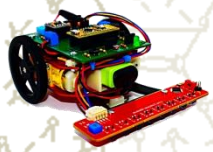




# Разширителна платка

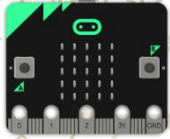


SOCIETY  
ROBOTIC

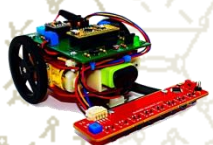
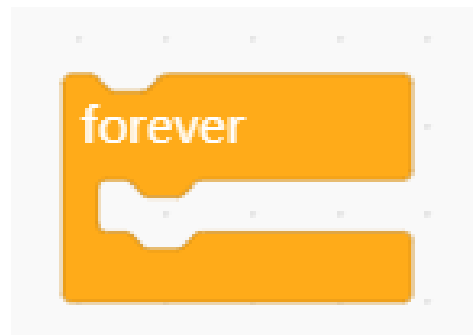




# Цикъл While



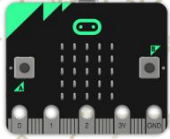
- Цикълът **WHILE** не се използва за определен брой повторения, а за повторения, докато дадено условие е изпълнено;
- В **Arduino** често се използва за създаване на безкраен цикъл, като по този начин се симулира работата на микроконтролера;



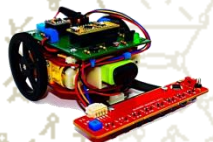
SOCIETY  
ROBOTICS



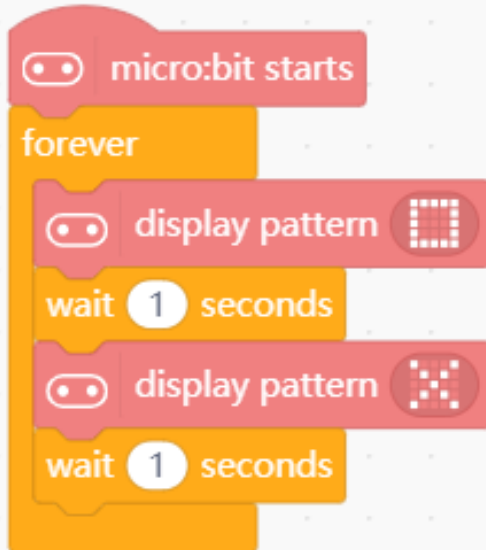
Society of Robotics



SOCIETY  
ROBOTIC



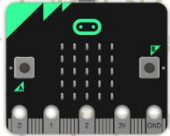
While\_1



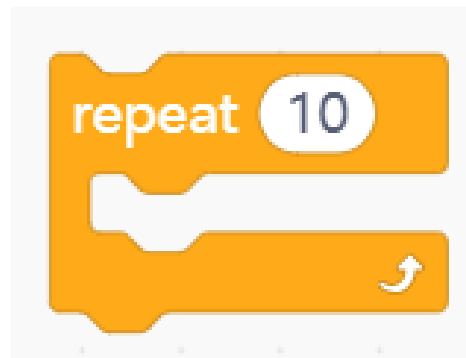
```
1 # MindPlus
2 # microbit
3 from microbit import *
4
5
6 while True:
7     display.show(Image("99999:90009:90009:90009:99999"))
8     sleep(1*1000)
9     display.show(Image("90009:09090:00900:09090:90009"))
10    sleep(1*1000)
11
```



# Цикъл Repeat

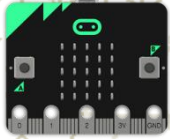


- Цикълът **REPEAT** (цикъл **FOR**) представлява програмна конструкция, която съдържа в себе си код, изпълнението на който се повтаря определен брой пъти;
- Цикълът **FOR** е цикъл с предварително определен брой на повторенията;

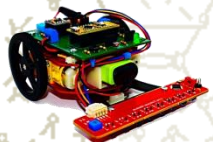




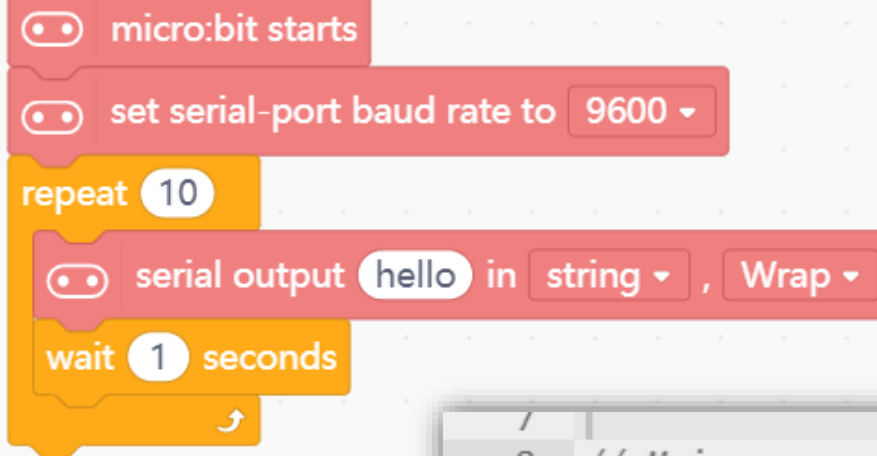
Society for Robotics



SOCIETY  
ROBOTIC



# Repeat\_2

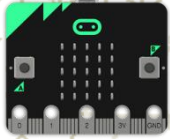


```
1 //  
2  
3  
4  
5  
6  
7  
8 // Main program start  
9 void setup() {  
10     Serial.begin(9600);  
11     for (int index = 0; index < 10; index++) {  
12         Serial.println("hello");  
13         delay(1000);  
14         yield();  
15     }  
16 }  
17 void loop() {  
18  
19 }  
20
```

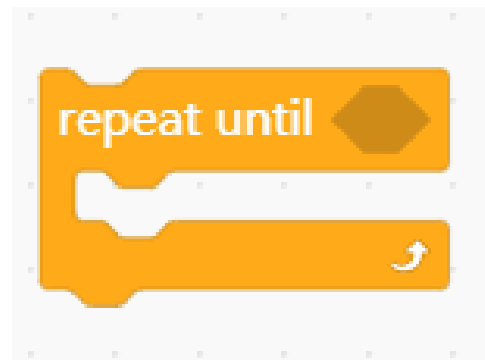




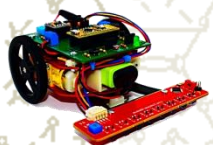
# Цикъл Repeat ..Until



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;



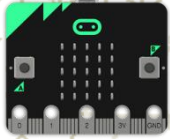
SOCIETY  
ROBOTICS



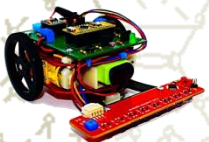




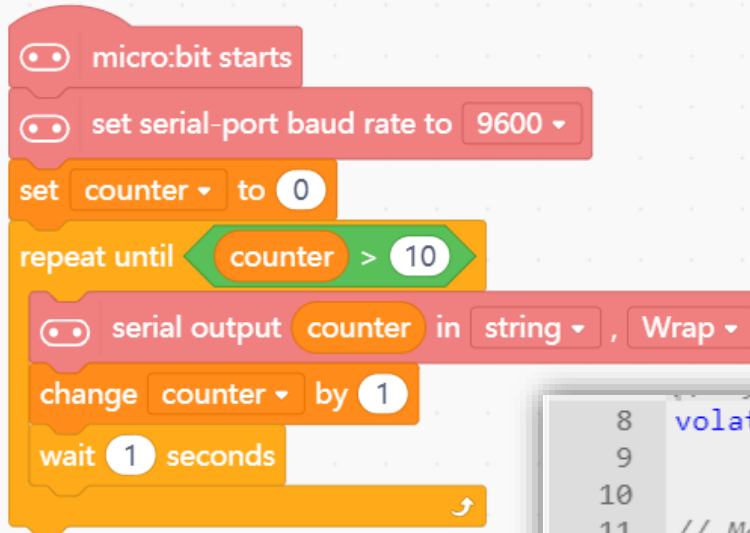
Society of Robotics



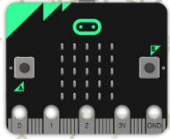
SOCIETY OF ROBOTICS



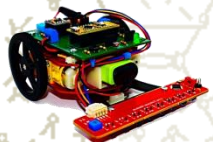
## Repeat \_Until\_3



```
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



Брояч

counter\_4

```
micro:bit starts
set counter to 0
set countingDown to 0
forever
  display counter
  if countingDown = 1 then
    wait 1 seconds
    change counter by -1
    if counter <= 0 then
      set countingDown to 0
      pin LS(V2) play sound RINGTONE
```

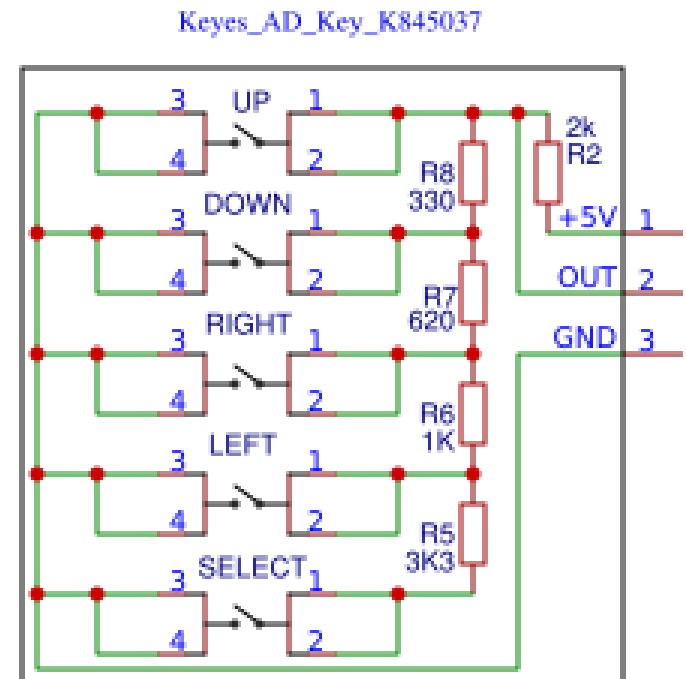
```
when button A pressed
  change counter by 1
```

```
when button B pressed
  set countingDown to 1
```



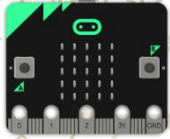
# KeyPad shield

Аналоговият **keypad shield** представлява делители на напрежения, свързани към бутони и организирани в един изход;

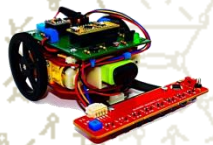




Society of Robotics



SOCIETY OF ROBOTICS



## keypad\_shield\_5

micro:bit starts

forever

set button\_key to read analog pin P2

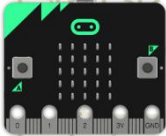
serial output button\_key in string , Wrap

wait 0.02 seconds

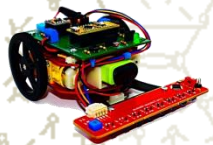
```
8  volatile float mind_n_button_key;
9
10
11  // Main program start
12  void setup() {
13    Serial.begin(9600);
14  }
15  void loop() {
16    mind_n_button_key = (analogRead(2));
17    Serial.println(mind_n_button_key);
18    delay(20);
19  }
```



Abir



SOCIETY  
ROBOTIC



micro:bit starts

forever

set button\_key to read analog pin P2

if button\_key < 10 then

display pattern 1

else if button\_key > 140 and button\_key < 155 then

display pattern 2

else if button\_key > 330 and button\_key < 345 then

display pattern 3

else if button\_key > 500 and button\_key < 530 then

display pattern 4

else if button\_key > 740 and button\_key < 760 then

display pattern 5

else

clear all dot matrixes

+

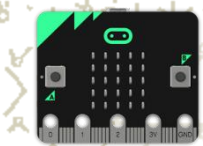
serial output button\_key in string , Wrap

wait 0.05 seconds

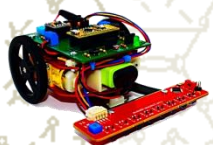
keypad\_shield\_5a



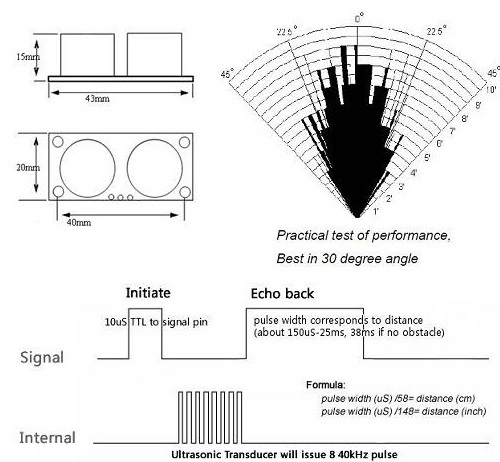
# Ултразвуков сензор



SOCIETY  
ROBOTICS



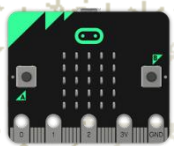
Ултразвуков сензор за разстояние HC-SR04. Намира приложение в проекти, където е необходимо отчитане на разстояние, избягване или откриване на обекти и др. Захранва се с напрежение 3-5V и консумира приблизително 6mA ток.



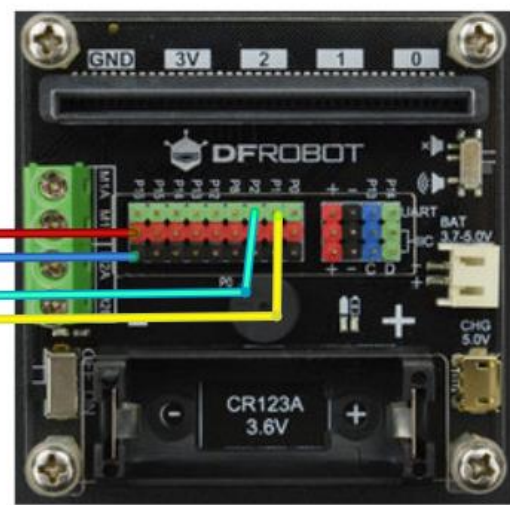
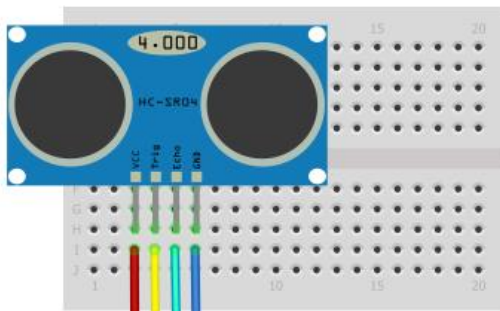
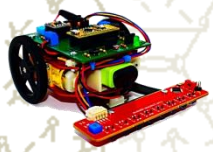




Society of Robotics



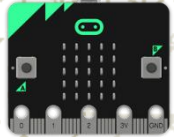
SOCIETY OF ROBOTICS

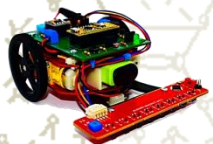






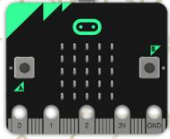
# Аналогов ултразвуков сензор

- 
- Ултразвуковият сензор е устройство, което може да измерва разстоянието до обект с помощта на звукови вълни;
  - DFRobot URM09** е ултразвуков сензор, специално проектиран за приложения с бързо измерване и избягване на препятствия. Неговата честота на измерване може да достигне до **30Hz**. Сензорът има аналогов изход. Може да осигури точно измерване на разстояние в рамките на **500 cm**;

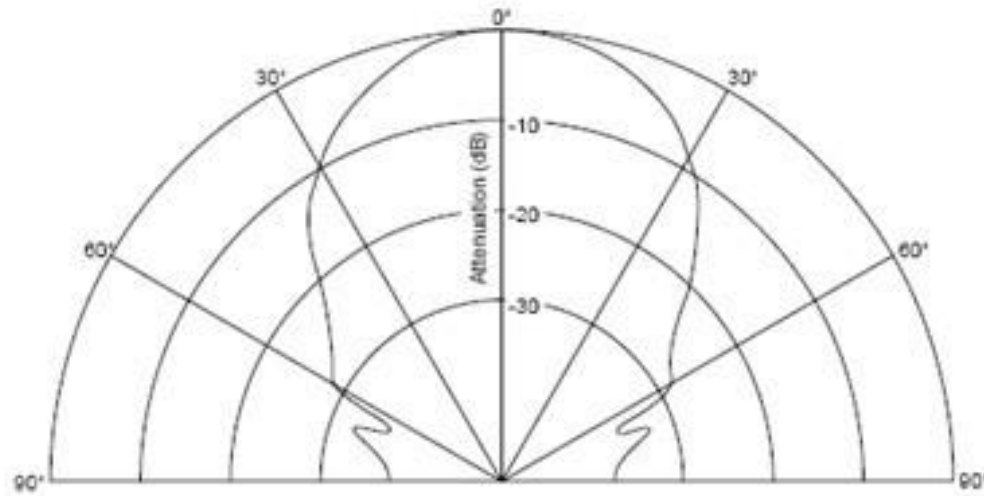
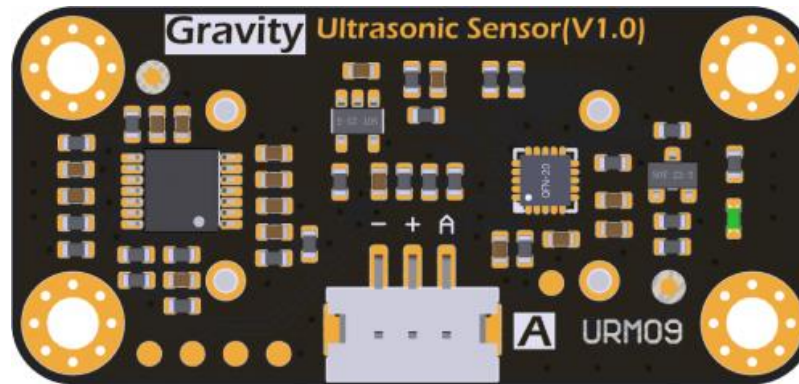
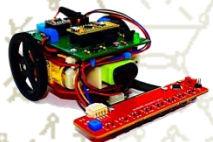


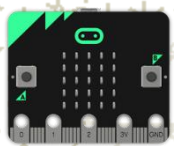


*Society of Robotics*

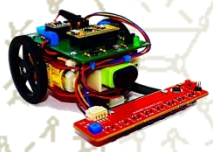


**SOCIETY OF ROBOTICS**





SOCIETY OF ROBOTICS



## Select Sensor

uator

Communication

Display

Function

Internet

User-Ext

A high accuracy ambient light  
digital 16-bit resolution  
sensor



SEN0248

**BME680 environmental se**  
VOC (volatile organic  
compounds), temperature,  
humidity and air pressure can

A high accuracy ambient light  
digital 16-bit resolution  
sensor



SEN0304

**I2C ultrasonic ranging sen**  
Ultrasonic sensors designed  
for rapid ranging or obstacle  
avoidance applications

Accuracy  $\pm 1\text{Pa}$ , high precision  
and high stability air pressure  
sensor

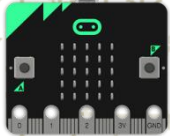


SEN0307

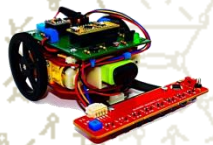
**Analog ultrasonic ranging**  
Open dual probe analog  
ultrasonic ranging module



Society of Robotics



SOCIETY OF ROBOTICS



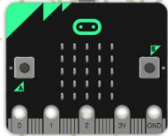
## ultrasonic\_6

```
micro:bit starts
forever
  set distance to Read P2 analog ultrasonic wave(cm)
  serial output Дистанцията е: in string , No-Wrap
  serial output distance in string , Wrap
  wait 0.1 seconds
```

```
6 #include <DFRobot_Ultrasonic.h>
7
8 // Dynamic variables
9 volatile float mind_n_distance;
10 // Create an object
11 DFRobot_Ultrasonic ultra2;
12
13
14 // Main program start
15 void setup() {
16   Serial.begin(9600);
17   ultra2.begin(2);
18 }
19 void loop() {
20   mind_n_distance = ultra2.getDistanceCm();
21   Serial.print("Дистанцията е: ");
22   Serial.println(mind_n_distance);
23   delay(100);
24 }
```

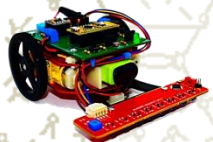


# Сервомотор SG90



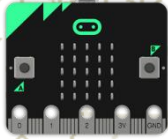
Аналоговият сервомотор SG90 е аналогов сервомотор с въртящ момент от 1.8 кг, с напрежение 4.8V и диапазон на въртене 180°. Обикновено сервомоторите са комбинация от четири неща - обикновен DC двигател, комплект предавки, потенциометър и контролна верига. Сред тези четири неща потенциометърът действа като позиционен сензор. В резултат на това могат да бъдат контролирани много прецизно;

SOCIETY  
ROBOTIC-

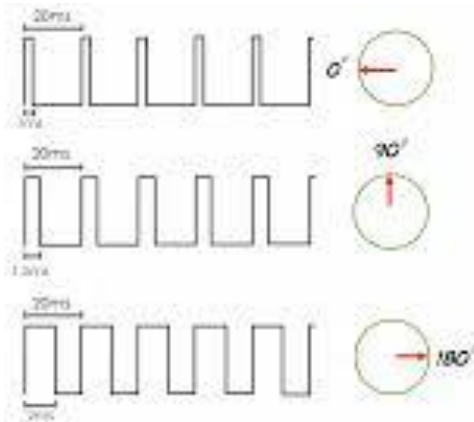
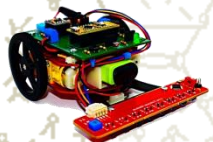




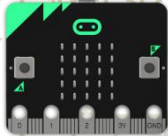
Abhishek



SOCIETY  
ROBOTIC







SOCIETY  
ROBOTIC

← Back

Select Actuator

Board

Kit

Shield

Sensor

Actuator

Communication

Display

Fun

Can't find what you want? [Click here](#) to find more



SER0006

**Micro Servo**

Micro Servo



SER0043|SER0035

**360° Micro Servo**

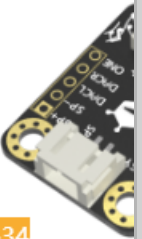
Control speed and direction



DFR0299

**DFPlayer MP3 Module**

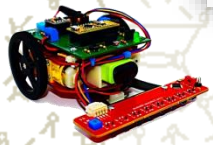
MP3 Player Module



DFR0534

**Serial MP3**

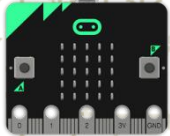
Serial MP3 v



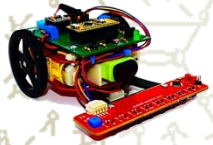




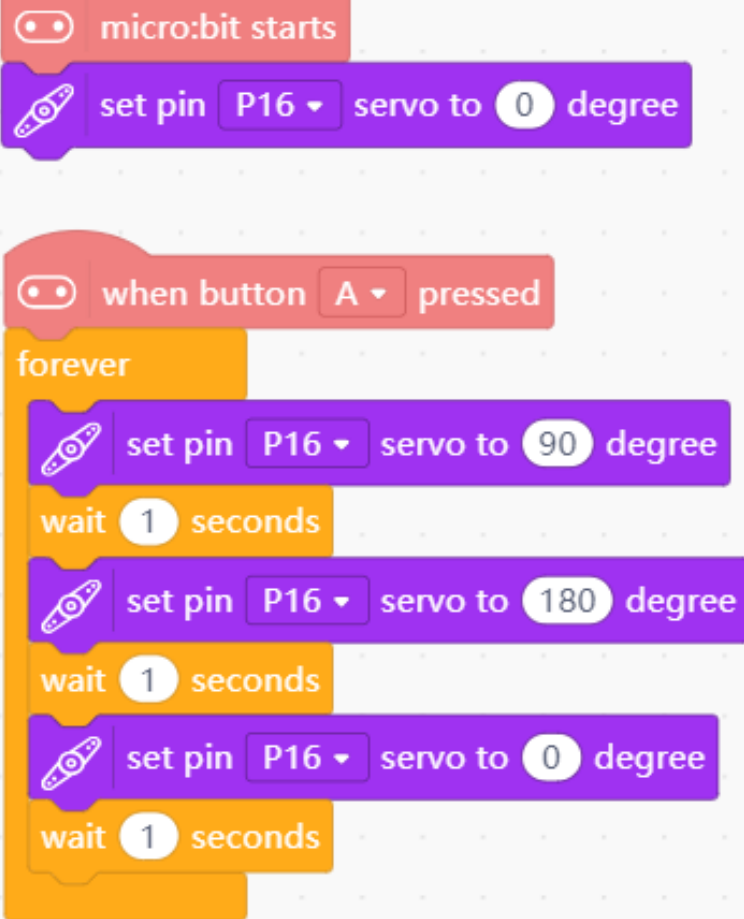
Society of Robotics



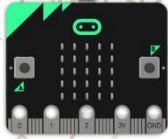
SOCIETY OF ROBOTICS



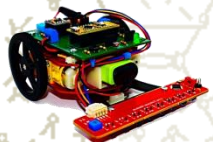
## servo\_motor\_7



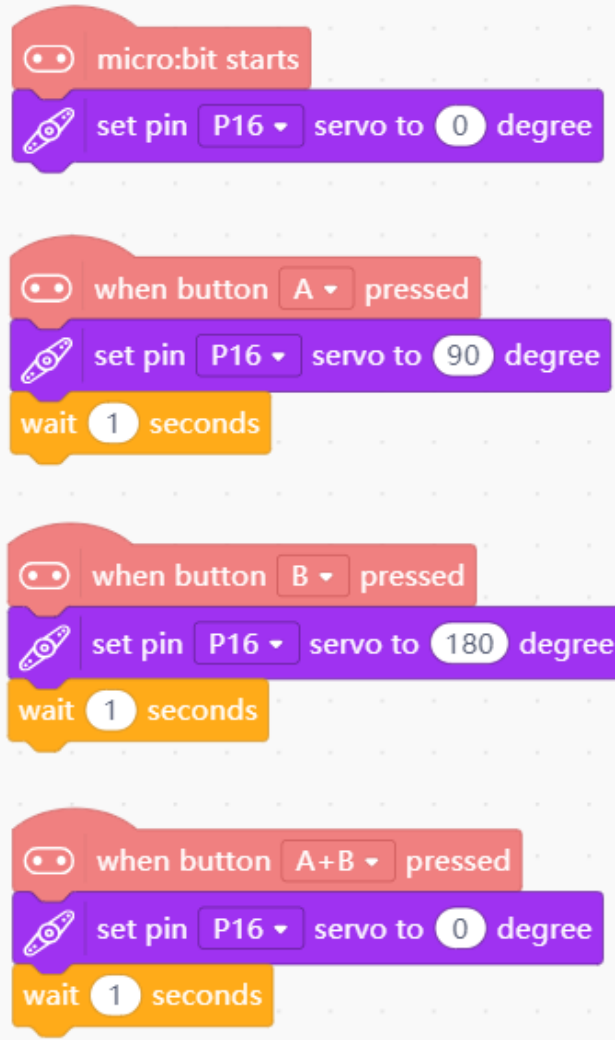
```
6 #include <DFRobot_Servo.h>
7 // Function declaration
8 void buttonACallback();
9 // Create an object
10 Servo servo_16;
11
12
13 // Main program start
14 void setup() {
15     servo_16.attach(16);
16     onEvent(ID_BUTTON_A, PRESS, buttonACallback);
17     servo_16.angle(abs(0));
18 }
19 void loop() {
20 }
21
22
23 // Event callback function
24 void buttonACallback() {
25     while (1) {
26         servo_16.angle(abs(90));
27         delay(1000);
28         servo_16.angle(abs(180));
29         delay(1000);
30         servo_16.angle(abs(0));
31         delay(1000);
32         yield();
33     }
34 }
35
```



SOCIETY  
ROBOTIC--



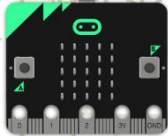
## servo\_motor\_buttonAB\_7a



```
6 #include <DFRobot_Servo.h>
7 // Function declaration
8 void buttonACallback();
9 void buttonBCallback();
10 void buttonABCallback();
11 // Create an object
12 Servo servo_16;
13
14
15 // Main program start
16 void setup() {
17     servo_16.attach(16);
18     onEvent(ID_BUTTON_A, PRESS, buttonACallback);
19     onEvent(ID_BUTTON_B, PRESS, buttonBCallback);
20     onEvent(ID_BUTTON_AB, PRESS, buttonABCallback);
21     servo_16.angle(abs(0));
22 }
23 void loop() {
24 }
25
26 // Event callback function
27 void buttonACallback() {
28     servo_16.angle(abs(90));
29     delay(1000);
30 }
31 void buttonBCallback() {
32     servo_16.angle(abs(180));
33     delay(1000);
34 }
35 void buttonABCallback() {
36     servo_16.angle(abs(0));
37     delay(1000);
38 }
39 }
```

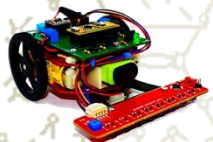


# Сензор DHT 11



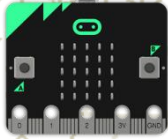
Сензорът за температура и влажност DHT11 разполага със сензор за температура и влажност с калибриран цифров сигнален изход.

Този сензор включва измерване на влажност от резистивен тип компонент и компонент за измерване на температура NTC и се свързва към 8-битов микроконтролер, предлагащ отлично качество, бърза реакция, защита срещу смущения, способност и рентабилност.

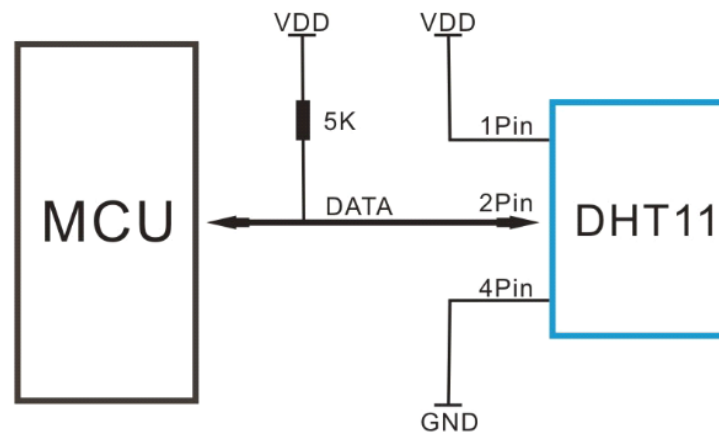
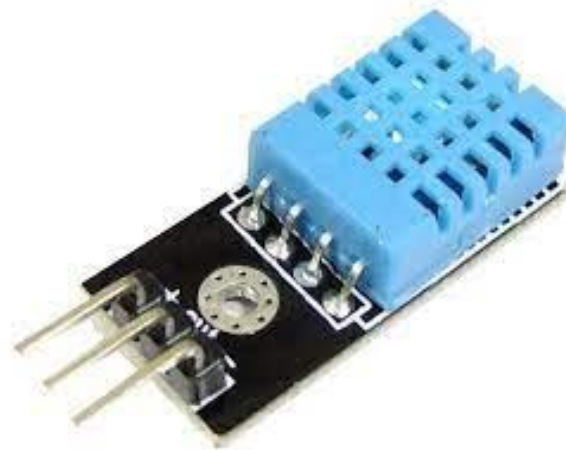
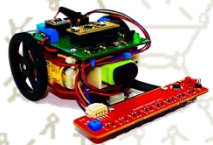


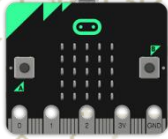


*Abhishek*

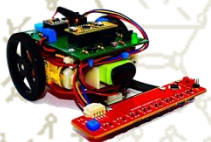


**SOCIETY  
ROBOTIC**





SOCIETY  
ROBOTIC




[← Back](#)[Select Sensor](#)

[Board](#)[Kit](#)[Shield](#)[Sensor](#)[Actuator](#)[Communication](#)[Display](#)[F](#)

Can't find what you want? [Click here](#) to find more


Loaded:



DFR0067|SEN0137

**DHT11/22 Temperature and Humidity sensor**


Detect environment temperature and humidity



SEN0307

**Analog ultrasonic ranging module**

Open dual probe analog ultrasonic ranging module



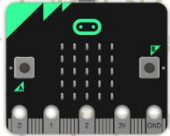
DFR0026

**Analog Ambient Light Sensor**

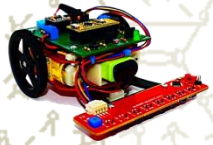
Detect the ambient light density



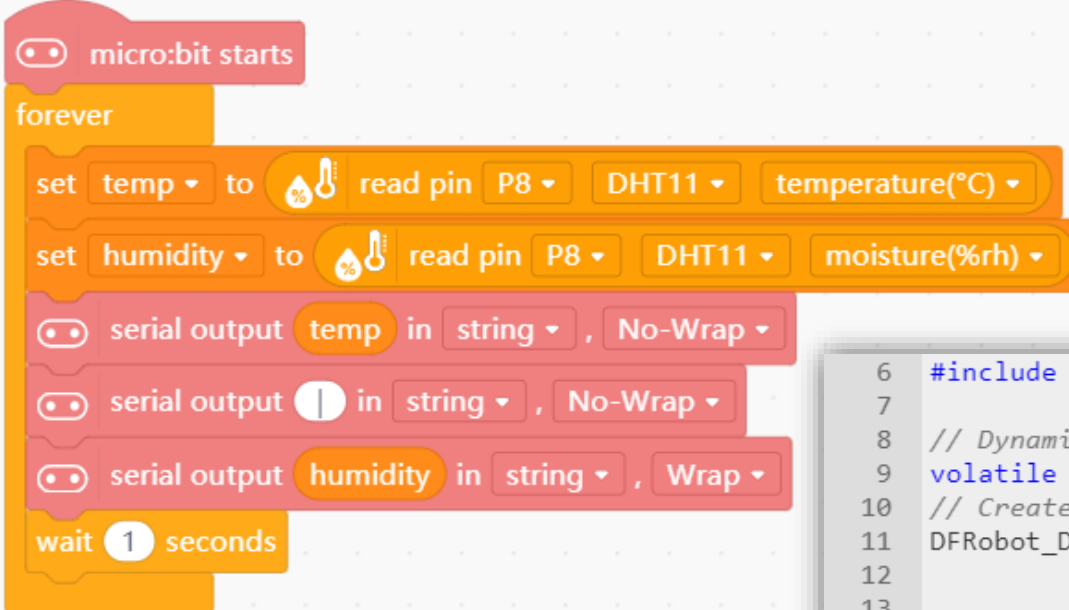
Society for Robotics



SOCIETY  
ROBOTIC-



# temp\_hum\_DHT11\_8

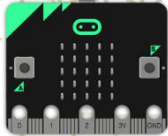


```
6 #include <DFRobot_DHT.h>
7
8 // Dynamic variables
9 volatile float mind_n_temp, mind_n_humidity;
10 // Create an object
11 DFRobot_DHT dht11_8;
12
13
14 // Main program start
15 void setup() {
16   Serial.begin(9600);
17   dht11_8.begin(8, DHT11);
18 }
19 void loop() {
20   mind_n_temp = dht11_8.getTemperature();
21   mind_n_humidity = dht11_8.getHumidity();
22   Serial.print(mind_n_temp);
23   Serial.print(" | ");
24   Serial.println(mind_n_humidity);
25   delay(1000);
26 }
```





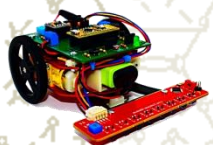
# Сензор DS18B20



Цифровият термометър DS18B20 осигурява 9-битов до 12-битови измервания на температурата по Целзий и има алармена функция с енергонезависима горна част, програмируема от потребителя и по-ниски тригерни точки.

DS18B20 комуникира през 1-Wire шина, която по дефиниция изисква само една линия за данни (и земя) за комуникация с микроконтролера;

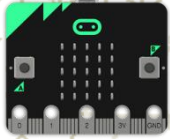
SOCIETY  
ROBOTIC-



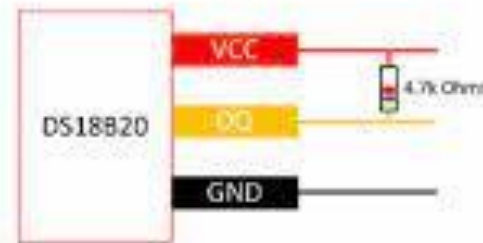
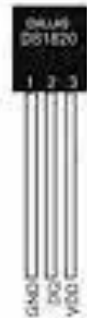
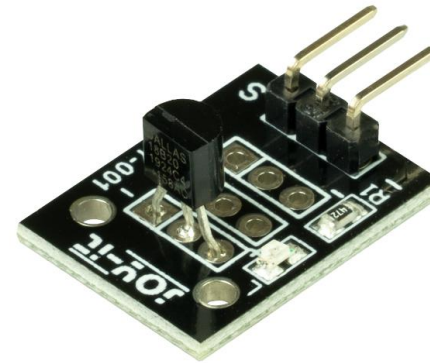
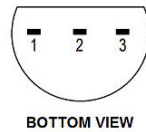
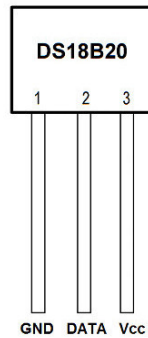
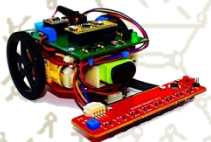


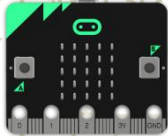


*Society of Robotics*

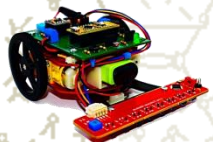


**SOCIETY OF ROBOTICS**





SOCIETY  
ROBOTIC



Select Sensor

Sensor

Actuator


Communication

Display

Function


Internet

User-Ext




**Ultrasonic Sensor**  
Distance detection  
range 2~800cm,  
compatible with urm and

DFR0023



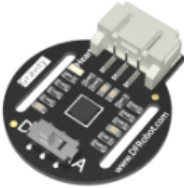
**Analog LM35 Temperature**  
LM35-based semiconductor  
temperature sensor with the  
range 0~100°C

DFR0024|KIT0021|DFR0198



**DS18B20 Temperature Sen**  
Detect ambient temperature  
with large range of  
-55~+125°C

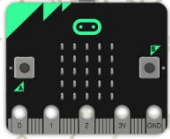
SEN0203



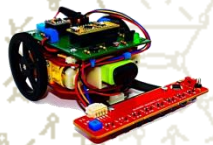
**Heart Rate Monitor Senso**  
Mini heart rate sensor with  
analog pulse and digital  
square wave output modes



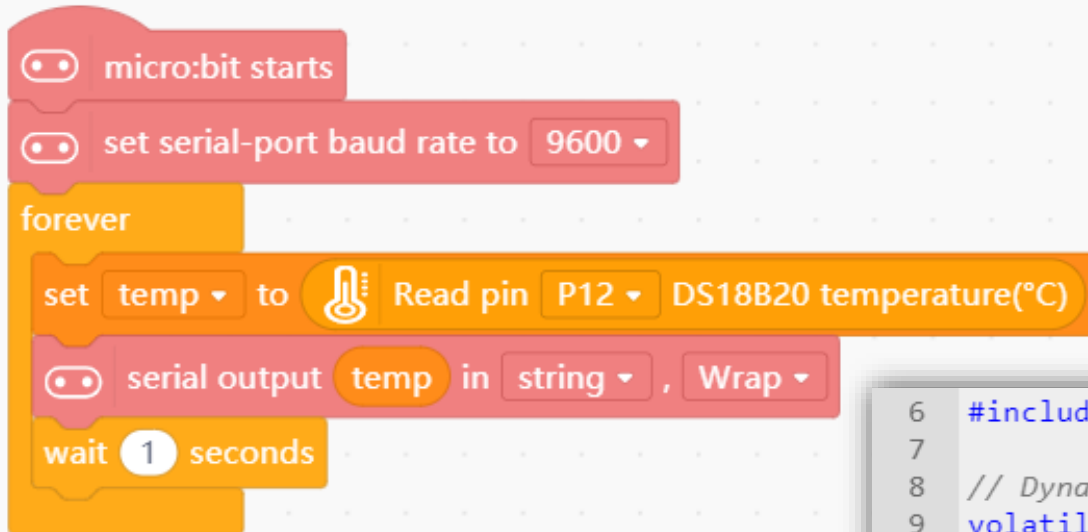
Society of Robotics



SOCIETY OF ROBOTICS



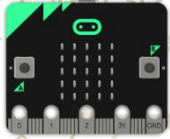
temp\_sensDS18B20\_9



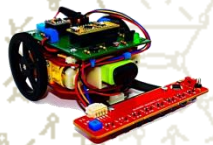
```
6 #include <DFRobot_DS18B20.h>
7
8 // Dynamic variables
9 volatile float mind_n_temp;
10 // Create an object
11 DFRobot_DS18B20 ds18b20_12;
12
13
14 // Main program start
15 void setup() {
16   ds18b20_12.begin(12);
17   Serial.begin(9600);
18 }
19 void loop() {
20   mind_n_temp = ds18b20_12.getTempC();
21   Serial.println(mind_n_temp);
22   delay(1000);
23 }
```



# Акселерометър



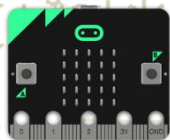
**SOCIETY  
ROBOTIC**



Weightless  
State



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



Да предположим, че кубът е в космоса, където всичко е в безтегловно състояние, топката просто ще се носи в средата на куба.

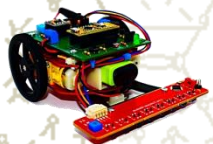


Сега нека си представим, че всяка стена представлява определена ос.



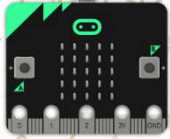
Ако внезапно преместим кутията наляво с ускорение  $1g$  (една G-сила  $1g$  е еквивалентна на гравитационно ускорение  $9,8 \text{ m/s}^2$ ), без съмнение топката ще удари стената X. Ако измерим силата, която топката прилага стената X, можем да получим изходна стойност от  $1g$  по оста X.

SOCIETY  
ROBOTIC

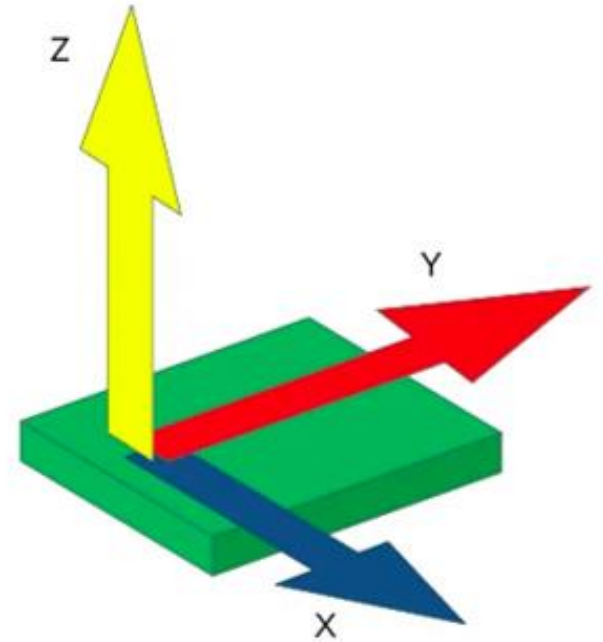
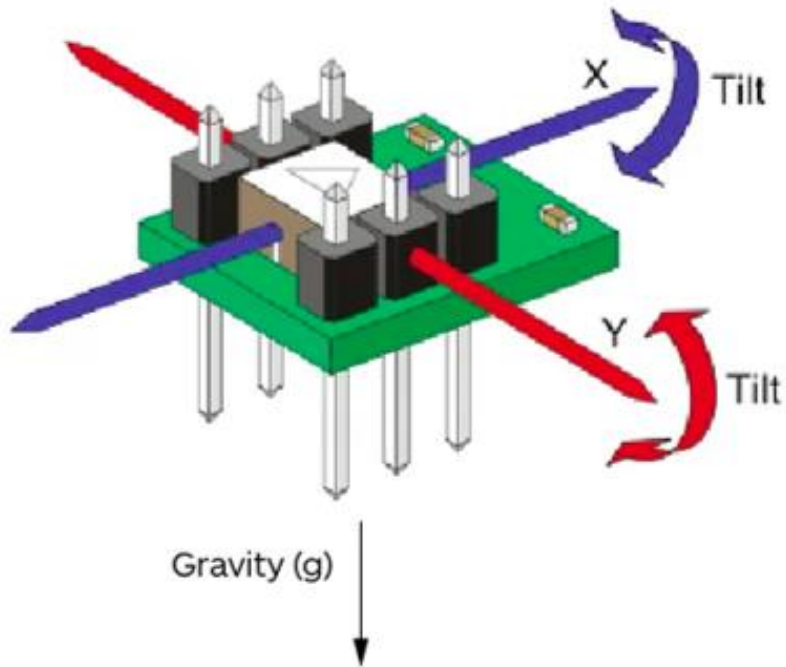
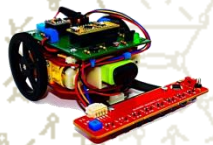




Abir



SOCIETY  
ROBOTIC

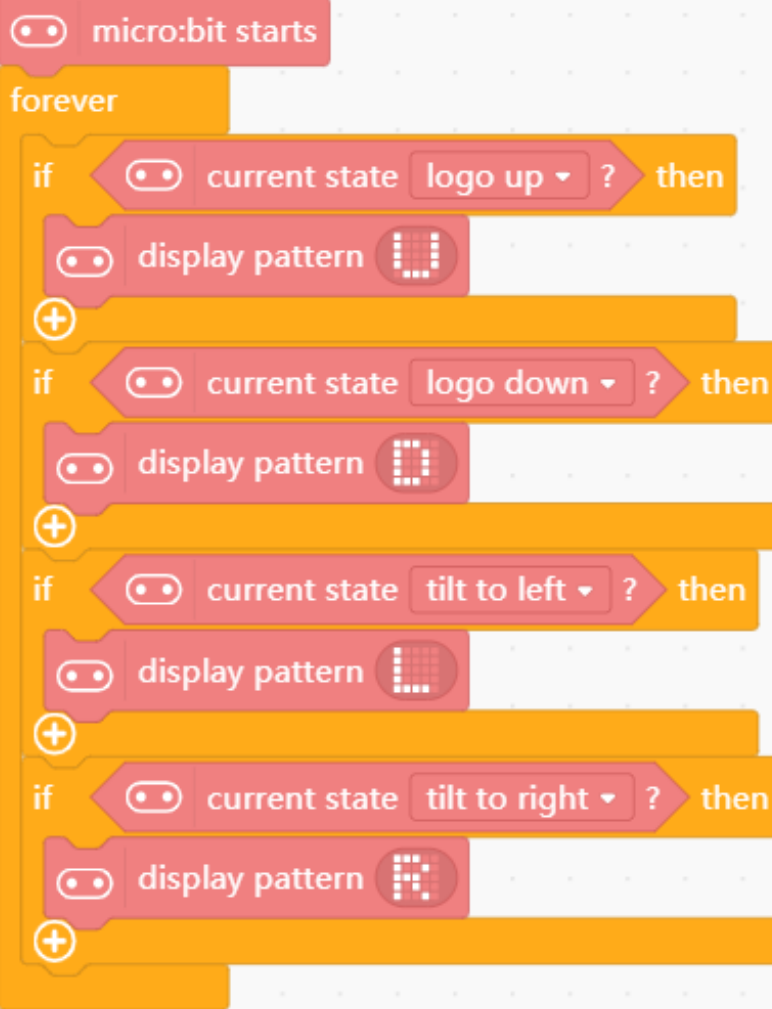
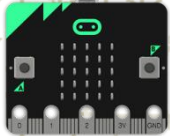






# Ориентация лого

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 }
```

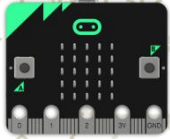






# Ускорения по осите

accelerXYZ\_11

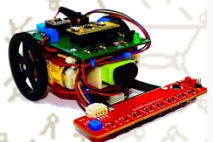


```
micro:bit starts
set serial-port baud rate to 9600

forever
  set aX to read acceleration(m-g) x
  set aY to read acceleration(m-g) y
  set aZ to read acceleration(m-g) z
  serial output aX in string, No-Wrap
  serial output | in string, No-Wrap
  serial output aY in string, No-Wrap
  serial output | in string, No-Wrap
  serial output aZ in string, No-Wrap
  serial output | in string, Wrap
  wait 0.5 seconds
```

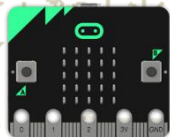
```
6 #include <Microbit_Sensors.h>
7
8 // Dynamic variables
9 volatile float mind_n_aX, mind_n_aY, mind_n_aZ;
10
11
12 // Main program start
13 void setup() {
14   Serial.begin(9600);
15 }
16 void loop() {
17   mind_n_aX = (Sensors.acceleration(Sensors.X));
18   mind_n_aY = (Sensors.acceleration(Sensors.Y));
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 }
```

SOCIETY  
ROBOTIC

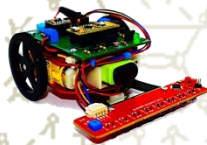




*Abriel*



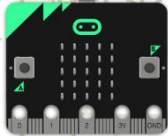
**SOCIETY  
ROBOTIC**



# УПРАЖНЕНИЕ

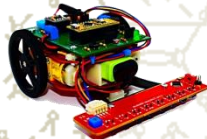


# Задача 1



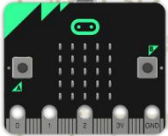
- ☀ Да се направи програма, която измерва температурата със сензор DS18B20 и ако е по-малка от **32 градуса** да извежда символ „√“ на LED дисплея, а ако е по-голяма от **32 градуса** да се извежда символ „!“;
- ☀ Да се изведе на терминал текущата стойност на температурата;

SOCIETY  
ROBOTICS

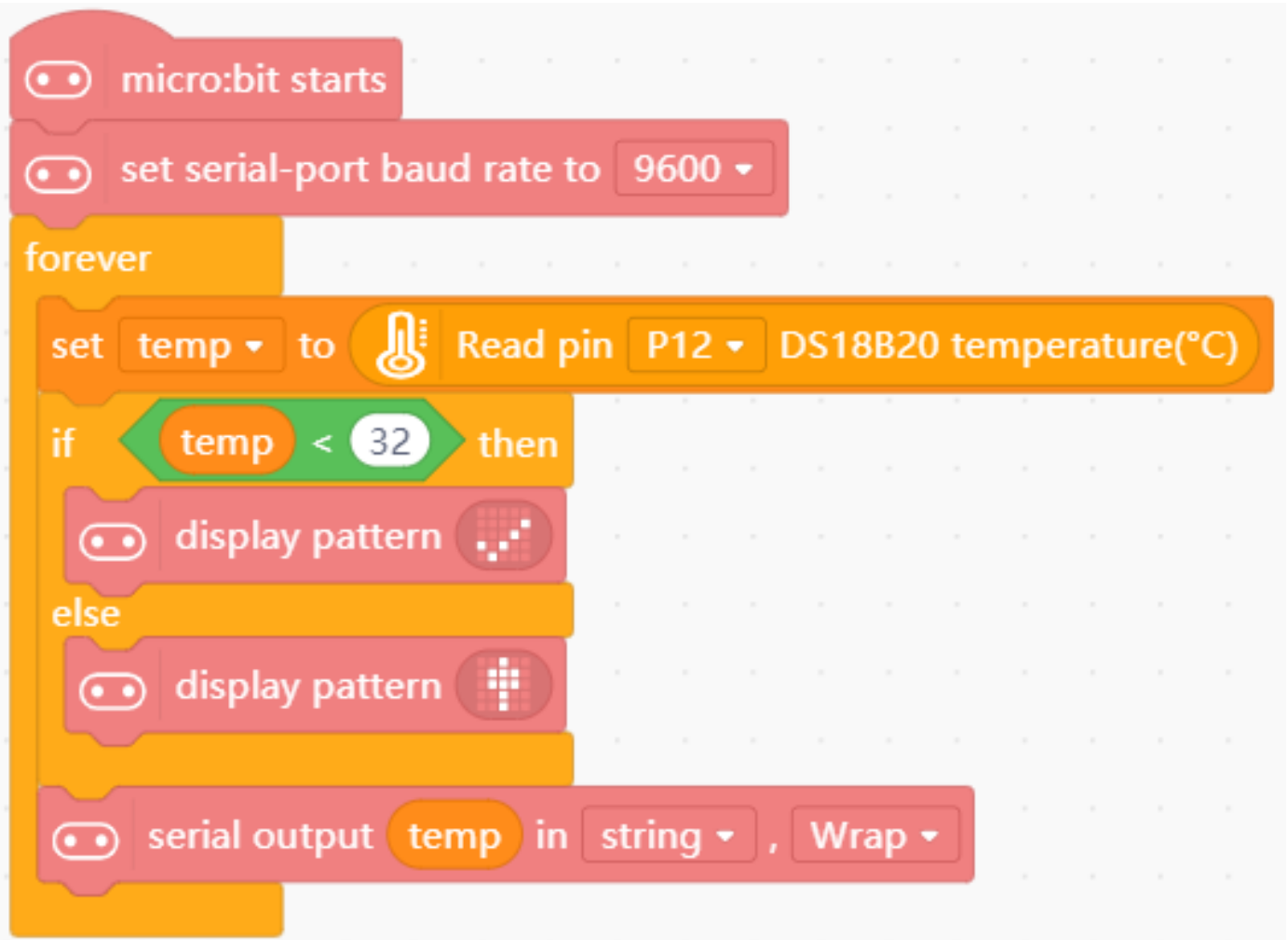
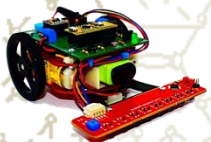




Society for Robotics

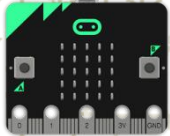


SOCIETY  
ROBOTIC

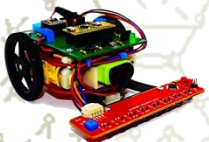


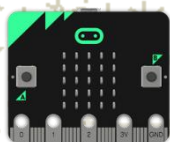


## Задача 2

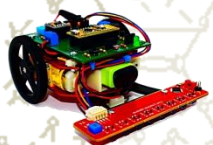


- Да се направи програма, която в зависимост от вертикалното положение на микроконтролера завърта сервомотора на **0 градуса** или на **180 градуса** и на LED дисплея да се визуализира стрелка с посоката на накланяне на микроконтролера;
- При стартиране сервомотора да се позиционира на **90 градуса**, което да отговаря на вертикално положение;

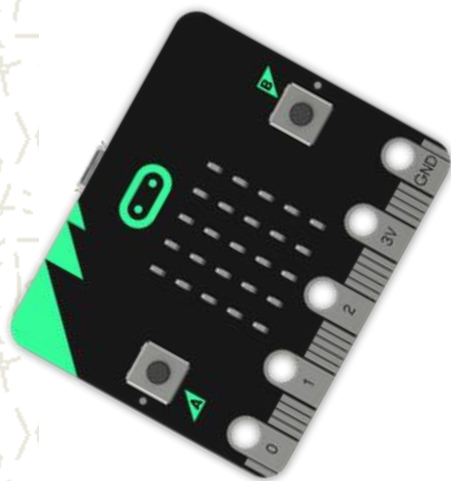




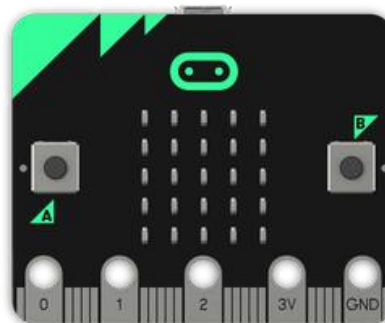
**SOCIETY  
ROBOTIC**



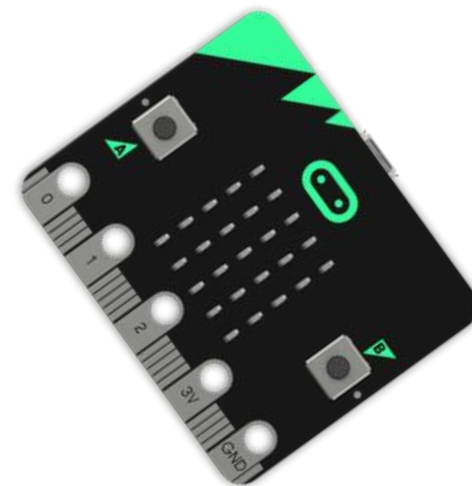
Наклонено  
наляво



Вертикално  
положение



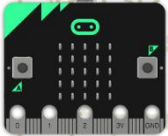
Наклонено  
надясно



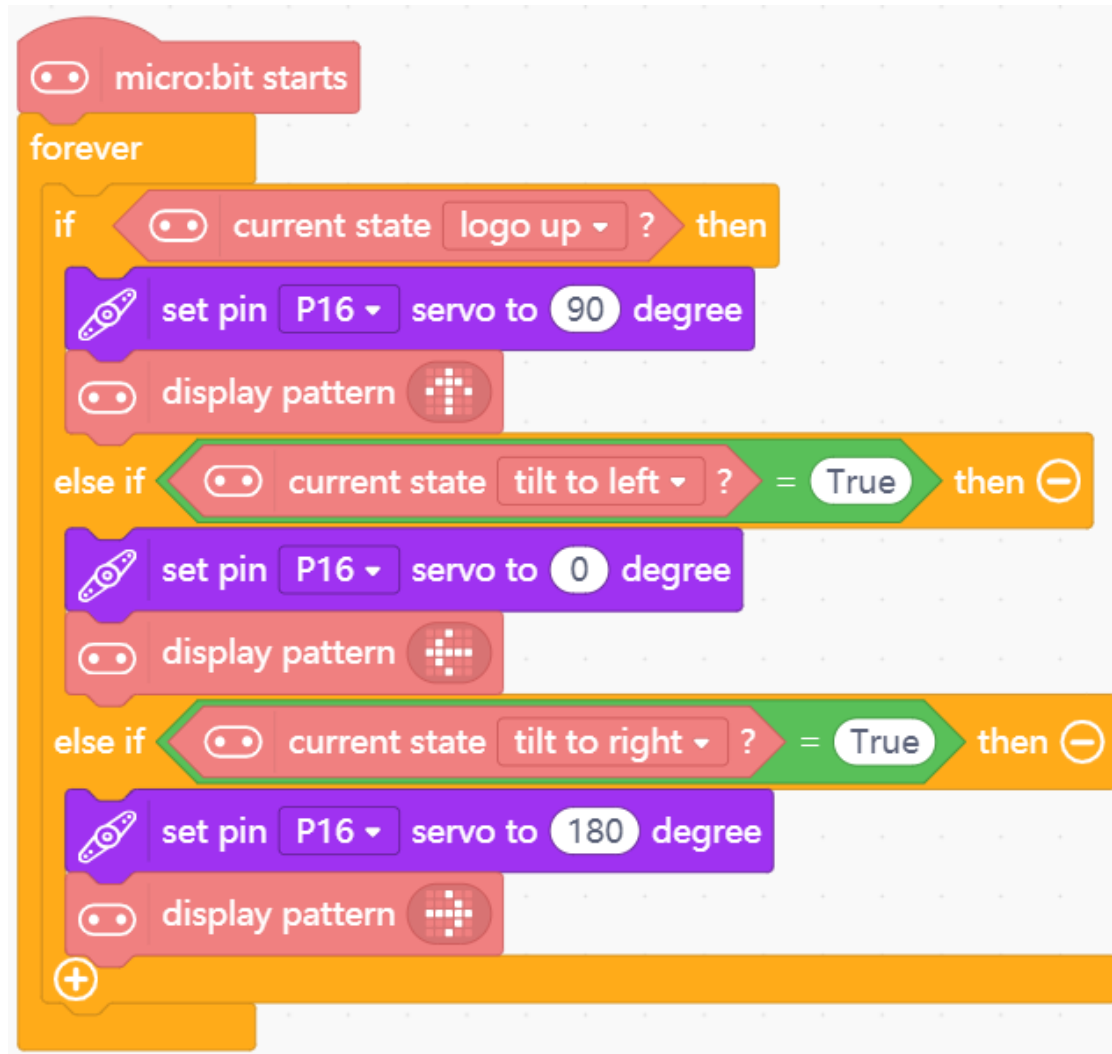
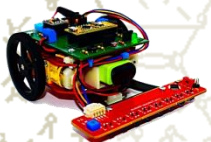




Abir



SOCIETY  
ROBOTIC--





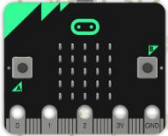
## Задача 3

Да се направи програма, която засича разстоянието до препятствие с ултразвуков аналогов сензор и ако разстоянието е по-малко от зададеното по схемата да се възпроизведе определен звуков сигнал;

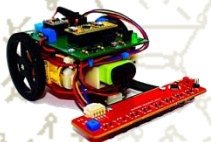
<b>Дистанция &lt; 25</b>	Звуков сигнал с висока честота на прекъсване
<b><math>25 \leq</math> Дистанция &lt; 35</b>	Звуков сигнал със средна честота на прекъсване
<b><math>35 \leq</math> Дистанция &lt; 50</b>	Звуков сигнал с ниска честота на прекъсване



Society of Robotics



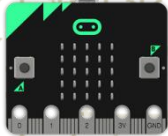
SOCIETY OF ROBOTICS



```
micro:bit starts
set serial-port baud rate to 9600
forever
  set distance to Read P1 analog ultrasonic wave(cm)
  if distance < 25 then
    pin P0 play note Middle F/F4 for 1/4 beat
  else if distance >= 25 and distance < 35 then
    pin P0 play note Middle F/F4 for 1/2 beat
  else if distance >= 35 and distance < 50 then
    pin P0 play note Middle F/F4 for 1 beat
  else
    stop background playback
    serial output distance in string, Wrap
  wait 0.5 seconds
```



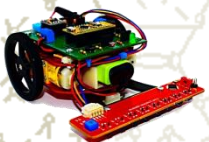
## Задача 4



☛ Да се направи програма, която използва аналогов ултразвуков сензор и светофарна уредба;

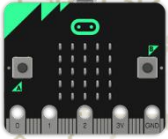
☛ При засичане на предмет на разстояние **по-малко от 30 см**, светофарната уредба да свети червено, ако предмета е на разстояние **от 30 см до 45 см**, да свети жълто, а ако предметът е на разстояние **по-голямо от 45 см**, да свети зелено;

SOCIETY  
ROBOTICS

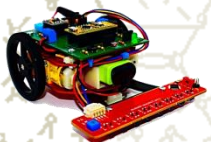




Society of Robotics



SOCIETY OF ROBOTICS



micro:bit starts

forever

set distance to read analog pin P1

if distance <= 30 then

digital pin P14 output HIGH

digital pin P15 output LOW

digital pin P16 output LOW

else if distance > 30 and distance <= 45 then

digital pin P14 output LOW

digital pin P15 output HIGH

digital pin P16 output LOW

else if distance > 45 then

digital pin P14 output LOW

digital pin P15 output LOW

digital pin P16 output HIGH

else

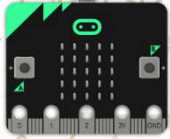
serial output distance in string , Wrap

wait 0.5 seconds



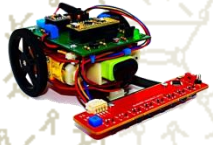


## Задача 5



- ☀ Да се направи програма, която използва сензор за интензитет на светлината LDR и сервомотор;
- ☀ При отчитане на нисък интензитет на светлината (смрачаване) сервомотора да се завърти на 180 градуса, а при отчитане на висок интензитет на светлината (изгрев) сервомотора да се завърти на 0 градуса;

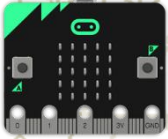
SOCIETY  
ROBOTICS



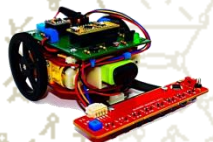




Society of Robotics



SOCIETY OF ROBOTICS



```
micro:bit starts
forever
  set light to read pin P2 Ambient light
  if light <= 100 then
    set pin P16 servo to 0 degree
  else
    set pin P16 servo to 180 degree
  wait 0.5 seconds
  serial output light in string , Wrap
```