Controllo dei Motori

Esercizio n°1

```
task main()
        {
                motor[motorC] = 50;
                motor[motorB] = 50;
                wait1Msec(4000);
                motor[motorC] = -50;
                motor[motorB] = 50;
                wait1Msec(800);
                motor[motorC] = 50;
                motor[motorB] = 50;
                wait1Msec(2000);
        }
```

Esercizio n°2

```
task main()
{
        int i = 0;                  // The variable 'i' is declared as an integer, and initialized to equal zero.
        while(i < 3)                // While 'i' is less than3:
        {
                motor[motorC] = 75;       // Motor C is run at a 75 power level.
                motor[motorB] = 0;        // Motor B is stopped.
                wait1Msec(750);             // The robot turns for 750 milliseconds before running further
code.
                motor[motorC] = -75;      // Motor C is run at a -75 power level.
                motor[motorB] = 0;        // Motor B is stopped.
                wait1Msec(750);             // The robot turns for 750 milliseconds before running further
code.
                motor[motorC] = 0;        // Motor C is stopped.
                motor[motorB] = 75;       // Motor B is run at a 75 power level.
                wait1Msec(750);             // The robot turns for 750 milliseconds before running further
code.
                motor[motorC] = 0;        // Motor C is stopped.
                motor[motorB] = -75;      // Motor B is run at a -75 power level.
                wait1Msec(750);             // The robot turns for 750 milliseconds before running further
code.
                i++;                      // The variable "i" is incremented (increased) by 1.
         }
}
```

Esercizio n°3

```
task main()

{

 nSyncedMotors = synchBC;      //motor B is the master, motor C is the slave
 nSyncedTurnRatio = -100;      //motors move in opposite directions of one another
 nMotorEncoder[motorB] = 0;  // Reset the Motor Encoder of Motor B.
 while(nMotorEncoder[motorB] < 760)
                               // While the Motor Encoder of Motor B has not yet reached 360 counts;
                               // (motor B turns one full wheel revolution)
 {   motor[motorB] = 30; }     //turn motor B on, which controls motor C at 30% power
   motor[motorB] = 0;          // turn the motors off.
 wait1Msec(3000);
}
```

____

Esercizio n° 4

/ /the program below uses the nMotorTargetEncoder function with synchronized motors

```
task main()
{
 nSyncedMotors = synchBC;                //motor B is the master, motor C is the slave
 nSyncedTurnRatio = -100;                //motors move in opposite directions of one another
 nMotorEncoder[motorB] = 0;              // clears the value of motorB's encoder
 nMotorEncoderTarget[motorB] = 760;  // sets a target of 360 degrees
 motor[motorB] = 30;                      //turns the motor on at 30% power
 while(nMotorRunState[motorB] != runStateIdle)          //while motorB is not in an idle state
 {  //continue to power motorB until the motor nMotorEncoderTarget position is reached }
 motor[motorB] = 0;                      // turn the motors off.
 wait1Msec(3000);
}
```

____

CONTROLLO DEL SENSORE AD ULTRASUONI

Esercizio n° 5

Il robot procede in avanti fino a quando non rileva un ostacolo, a questo punto si ferma:

```
#pragma config(Sensor, S4,    sonarSensor,       sensorSONAR)
                              // #pragma serve a configurare il sensore ad ultrasuoni alla porta di ingresso S4
task main()
{
 int distance_in_cm = 20;        // Create variable 'distance_in_cm' and initialize it to 20(cm).
  while(SensorValue[sonarSensor] > distance_in_cm)
                    // While the Sonar Sensor readings are less than the specified,  'distance_in_cm':
  {
    motor[motorB] = 35;          // Motor B is run at a 35 power level
    motor[motorC] = 35;          // Motor C is run at a 35 power level
  }
  motor[motorB] = 0;     // Motor B is stopped at a 0 power level
  motor[motorC] = 0;     // Motor C is stopped at a 0 power level
}
```

Esercizio n° 6

Il robot si muove in avanti sino a quandol’oggetto non raggiunge la distanza di 35cm

per invertire la marcia se l'oggetto viene più vicino rispetto alla distanza specificata

```
#pragma config(Sensor, S4,    sonarSensor,       sensorSONAR)
task main()
{
 int speed = 0;                    // Will hold the speed of the motors.
 int sonarValue = 0;               // Will hold the values read in by the Sonar Sensor.
 int distance = 35;                // Specified distance to be at 35 centimeters.
 while(true)
// (infinite loop, also represented by 'while(1)' or, if you are feeling devious, 'for(;;)' which is read as 'for
ever').
 {
   sonarValue = SensorValue(sonarSensor);          // Store Sonar Sensor values in 'sonarValue' variable.
   nxtDisplayCenteredTextLine(0, "Sonar Reading");               //Display Sonar Sensor values
   nxtDisplayCenteredBigTextLine(2, "%d", sonarValue);           // to LCD screen using %d.
   wait1Msec(100);                                          // Only update the screen every 100 milliseconds.
   speed = (SensorValue(sonarSensor) - distance);          // Variable 'speed' is set to the reading of the Sonar
                                                     //Sensor - some distance in centimeters (here we used 35cm).
if(speed > 100)
   {
        speed = 100;     // Check to see if calculated speed is greater than 100, if so make it 100.
   }
   nxtDisplayCenteredTextLine(5, "%d", speed);         /* Display variable 'speed' to the LCD.      */
   nxtDisplayCenteredTextLine(7, "Motor Speed");     /* (which is the current speed of the motors) */
   motor[motorC] = speed;                          // Set Motor C is run at a power level equal to 'speed'.
   motor[motorB] = speed;                          // Set motor B is run at a power level equal to 'speed'.
 }
}
```

CONTROLLO DEL SENSORE DI SUONO

Esercizio n° 7

```
#pragma config(Sensor, S1,     soundSensor,        sensorSoundDB)
                              //This program runs your robot forward until a loud noise is made.
task main()
{
          wait1Msec(1000);                  // Wait for 1 second to ignore initial readings of the Sound Sensor.
          while(SensorValue(soundSensor) < 70)             // While the Sound Sensor is less than 70 (quiet):
          {
                    motor[motorC] = 75;                   // Motor C is run at a 75 power level.
                    motor[motorB] = 75;                   // Motor B is run at a 75 power level.
          }
          motor[motorC] = 0;                          /* Otherwise, when loud noises are heard, Motor C */
          motor[motorB] = 0;                 /* and motor B stop.                        */
}
```

____

ESERCIZIO n°8

```
#pragma config(Sensor, S2,     soundSensor,        sensorSoundDB)
/* La velocità di movimento del robot dipende dal volume di rumore rilevato dal sensore           *|
Più forte è il suono, più velocemente il robot andrà..   */
task main()
{
  wait1Msec(1000);      // A one-second wait is required to cleanly initialize the Sound Sensor.
  while(true)                 // Infinite loop
  {
    motor[motorB] = SensorValue[soundSensor];    /* Motors B and C are run at a power level equal */
    motor[motorC] = SensorValue[soundSensor];    /* to the value read in by the Sound Sensor.     */
  }
}
```

____

CONTROLLO DEL SENSORE DI CONTATTO

ESERCIZIO n°9

```
// #pragma config(Sensor, S1,    touchSensor,        sensorTouch)

const  tSensors  touchSensor         = (tSensors) S1;  //sensorTouch
/*  Il robot procede in vanti sino a che il sensore di contatto non è urtato.
    Se il sensore Touch è urtato, il robot si annulla e stop..                        */
task main()
{
 while(SensorValue(touchSensor) == 0)            // While the Touch Sensor is inactive (hasn't been pressed):
 {
  motor[motorB] = 100;                       /* Run motors B and C forward */
  motor[motorC] = 100;                       /* with a power level of 100. */
 }
                                  // Otherwise (the touch sensor has been activated [pressed] ):
 motor[motorB] = -75;                        /* Run motors B and C backwards */
 motor[motorC] = -75;                /* with a power level of -75.   */

 wait1Msec(1000);                            // Wait 1000 milliseconds (1 second) before moving to further code.
}
```

ESERCIZIO n°10

```
#pragma config(Sensor, S1,    touchSensor,        sensorTouch)
task main()
{
 while(SensorValue(touchSensor) == 0)   // While the Touch Sensor is inactive (hasn't been pressed):
 {
  // DO NOTHING (wait for press)
 }

 while(SensorValue(touchSensor) == 1)   // While the Touch Sensor is active (pressed):
 {
  // DO NOTHING (wait for release)
 }
                                 // Otherwise (the touch sensor has been activated [pressed] ):
 motor[motorB] = 75;                 /* Run motors B and C forwards */
 motor[motorC] = 75;                 /* with a power level of 75.   */

 wait1Msec(1000);                  // Wait 1000 milliseconds (1 second) before moving to further code.
}
```

Esercizio n° 11

```
#pragma config(Sensor, S1,    touchRight,        sensorTouch)
#pragma config(Sensor, S2,    touchLeft,         sensorTouch)
/* Questo programma utilizza due sensori di contatto.
  Se  Touch Sensor di destra è urtato, il robot girera' sinistra, e poi continuera' a spostarsi
 in avanti. Allo stesso modo, se il Touch Sensor di sinistra  è colpito, il robot  girera' a destra, e poi
 continuera' in avanti. */
task main()
{
 int randTime;                    // Declare variable 'randTime' to hold a random amount of time later.

 wait1Msec(500);                    // Wait 500 milliseconds before running any further code.

 while(true)                      // Infinite loop (also represented by 'while(1)' and 'for(;;)' which is read as 'for
ever').
 {
  motor[motorC] = 75;               /* Motors A and B are run */
  motor[motorB] = 75;               /* at a power level of 75 */

  if(SensorValue(touchRight) == 1)     // If the Right Touch Sensor is bumped (equal to 1):
  {
   motor[motorC] = -75;                /* Motors A and B are run  */
   motor[motorB] = -75;                /* at a power level of -75 */
   wait1Msec(750);                    /* for 750 milliseconds.   */

   motor[motorC] = 75;                 // Motor C is run forward at a power level of 75.
   motor[motorB] = -75;                // Motor B is run backward at a power level of -75.
   randTime = random(2000);               // 'randTime' is set to a random integer between 0 and 2000.
   wait1Msec(randTime);                 // Wait 'randTime' amount of milliseconds.
  }

  if(SensorValue(touchLeft) == 1)      // If the Left Touch Sensor is bumped (equal to 1):
  {
   motor[motorC] = -75;                /* Motors A and B are run  */
   motor[motorB] = -75;                /* at a power level of -75 */
   wait1Msec(750);                    /* for 750 milliseconds.   */

   motor[motorC] = -75;                // Motor C is run backward at a power level of 75.
   motor[motorB] = 75;                 // Motor B is run forward at a power level of 75.
   randTime = random(2000);               // 'randTime' is set to a random integer bteween 0 and 2000.
   wait1Msec(randTime);                 // Wait 'randTime' amount of milliseconds.
  }
 }
}
```