```
task main()
{
    OnFwd(OUT_AC, 75);
    Wait(800);
    OnRev(OUT_C, 75);
    Wait(360);
    Off(OUT_AC);
}
```

Utilización de constantes

```
#define MOVE_TIME 1000
#define TURN_TIME 360
task main()
{
    OnFwd(OUT_AC, 75);
    Wait(MOVE_TIME);
    OnRev(OUT_C, 75);
    Wait(TURN_TIME);
    Off(OUT_AC);
}
```

Repeticiones

```
#define MOVE_TIME 500
#define TURN_TIME 500
task main()
{
    repeat(4)
        {
        OnFwd(OUT_AC, 75);
        Wait(MOVE_TIME);
        OnRev(OUT_C, 75);
        Wait(TURN_TIME);
        }
    Off(OUT_AC);
    }
```

```
#define MOVE_TIME 500
#define TURN_TIME 500
task main()
{
    repeat (4)
    {
        repeat(4)
```

```
        {
            OnFwd(OUT_AC, 75);
            Wait(MOVE_TIME);
            OnRev(OUT_C, 75);
            Wait(TURN_TIME);
        }
        Off(OUT_AC);
    }
}
```

Apartado 3

Sentencia repeat

```
#define TURN_TIME 360

int move_time;

task main()
{
    move_time = 200;
    repeat(50)
    {
        OnFwd(OUT_AC, 75);
        Wait(move_time);
        OnRev(OUT_C, 75);
        Wait(TURN_TIME);
        move_time += 200;
    }
    Off(OUT_AC);
}
```

Sentencia if-else

A veces queremos que una parte de nuestro programa se ejecute solamente en ciertas situaciones. En esos casos se usa la sentencia if. Vamos a ver un ejemplo. Vamos a modificar el programa con el que hemos estado trabajando, pero queremos que gire bien a la derecha o a la izquierda, y que haga esa elección de modo aleatorio. Elegiremos al azar un número que puede ser positivo o negativo, y si es positivo el robot girará a la derecha y si no, girará hacia la izquierda.

Si la condición entre paréntesis es cierta, se ejecuta la parte entre paréntesis, si no lo es, se ejecutará la parte detrás de la sentencia else.

Prestemos atención a la condición que hemos usado: Random() >= 0, esto significa que Random() debe ser igual o mayor que cero para que la condición sea cierta. Se pueden comparar los valores de otras maneras, aquí vemos las más importantes:

==	igual a
<	menor que
<=	menor o igual a
>=	mayor o igual a
¡=	no igual a

Se pueden combinar condiciones usando &&, que significa “y” o “||” que significa “o”. Veamos algunos ejemplos: true siempre cierto

Fíjate que la sentencia if tiene dos partes. La parte inmediatamente después de la condición, que se ejecuta cuando la condición es cierta, y la parte después del else, que se ejecuta si la condición es falsa.

La palabra clave else y la parte que le sigue son opcionales, de manera que puedes omitirlas si no hay nada que hacer en caso de que la condición sea falsa.

Output

```
#define MOVE_TIME 500
#define TURN_TIME 360
task main() {
  while (true) {
    OnFwd(OUT_AC, 75);
    Wait(MOVE_TIME);
    if (Random() >= 0) {
      OnRev(OUT_C, 75);
    } else {
      OnRev(OUT_A, 75);
    }
    Wait(TURN_TIME);
  }
}
```

Sentencia while

```
int move_time, turn_time;
task main()
{
    while(true)
    {
        move_time = Random(600);
        turn_time = Random(400);
        OnFwd(OUT_AC, 75);
        Wait(move_time);
        OnRev(OUT_A, 75);
        Wait(turn_time);
    }
}
```

Apartado 4

```
#define MOVE_TIME 500
#define TURN_TIME 360
task main()
    {
    while(true)
    {
        OnFwd(OUT_AC, 75);
        Wait(MOVE_TIME);
        if (Random() <= 0)
        {
        OnRev(OUT_A, 75);
        }
        else
        {
        OnRev(OUT_C, 75);
        }
        Wait(TURN_TIME);
    }
}
```

Bucles do-while

Hay otra estructura de control, la sentencia do. Tiene la siguiente forma:

Output

```
do {
  instrucciones;
}
while (condition);
```

Las instrucciones entre llaves que hay después del do se ejecutan mientras la condición sea cierta. La condición tiene la misma forma que el if. Éste es un ejemplo de un programa. El robot da vueltas de modo aleatorio durante 20 segundos y después se para.

Fíjate que la sentencia do es casi igual que la sentencia while. Pero en el while, la condición se comprueba antes de entrar en las instrucciones, mientras que en el do, se comprueba al final. En el caso del while puede ocurrir que las sentencias no se ejecuten nunca, pero en el do, se ejecutarán al menos una vez.

```
int move_time, turn_time, total_time;
task main()
{
    total_time = 0;
    do
    {
        move_time = Random(1000);
        turn_time = Random(1000);
```

```
        OnFwd(OUT_AC, 75);
        Wait(move_time);
        OnRev(OUT_C, 75);
        Wait(turn_time);
        total_time += move_time;
        total_time += turn_time;
    }
    while (total_time < 20000);
    Off(OUT_AC);
}
```

Fíjate que la sentencia do es casi igual que la sentencia while. Pero en el while, la condición se comprueba antes de entrar en las instrucciones, mientras que en el do, se comprueba al final. En el caso del while puede ocurrir que las sentencias no se ejecuten nunca, pero en el do, se ejecutarán al menos una vez.

Apartado 5

```
task main()
{
    SetSensor(IN_1,SENSOR_TOUCH);
    OnFwd(OUT_AC, 75);
    until (SENSOR_1 == 1);
    Off(OUT_AC);
}
```

```
task main()
{
    SetSensorTouch(IN_1);
    OnFwd(OUT_AC, 75);
    while (true)
    {
        if (SENSOR_1 == 1)
        {
        OnRev(OUT_AC, 75); Wait(300);
        OnFwd(OUT_A, 75); Wait(300);
        OnFwd(OUT_AC, 75);
        }
    }
}
```

Luz

```
#define SPEED 60
#define motoren OUT_BC
#define THRESHOLD 45
task main ()
```

```
{
    SetSensorLight(IN_3);
    OnFwd(motoren, SPEED);
    while(true)
    {
        if(SENSOR_3 < THRESHOLD)
        {
        Off(OUT_B);
        OnFwd(OUT_C, SPEED);
        }
        else
        {
        Off(OUT_C);
        OnFwd(OUT_B, SPEED);
        }
    }
}
```

Sonido

```
#define THRESHOLD 40
#define MIC SENSOR_2
task main()
{
    SetSensorSound(IN_2);
    while(true)
    {
        until(MIC > THRESHOLD);
        OnFwd(OUT_AC, 75);
        Wait(300);
        until(MIC > THRESHOLD);
        Off(OUT_AC);
        Wait(300);
    }
}`
```

Ultrasonidos

```
#define NEAR 15

task main()
{
    SetSensorLowspeed(IN_4);
    while(true)
    {
    OnFwd(OUT_AC,50);
    while(SensorUS(IN_4)>NEAR);
    Off(OUT_AC);
    OnRev(OUT_C,100);
```

```
    Wait(800);
    }
}
```

6. Subrutinas

```
sub turn_around(int pwr) //Anlegen einer Subroutine
{
    OnRev(OUT_C, pwr);
    Wait(900);
    OnFwd(OUT_AC, pwr);
}
task main()
{
    OnFwd(OUT_AC, 75);
    Wait(1000);
    turn_around(75); //Aufruf der Subroutine
    Wait(2000);
    turn_around(75); //Aufruf der Subroutine
    Wait(1000);
    turn_around(75); //Aufruf der Subroutine
    Off(OUT_AC);
}
```

7. Música

```
#define VOL 3
task main()
{
PlayToneEx(262,400,VOL,FALSE); Wait(500);
PlayToneEx(294,400,VOL,FALSE); Wait(500);
PlayToneEx(330,400,VOL,FALSE); Wait(500);
PlayToneEx(294,400,VOL,FALSE); Wait(500);
PlayToneEx(262,1600,VOL,FALSE); Wait(2000);
}
```

```
task music()
{
while (true)
{
PlayTone(262,400); Wait(500);
PlayTone(294,400); Wait(500);
PlayTone(330,400); Wait(500);
PlayTone(294,400); Wait(500);
}
}
```

```
task movement()
{
while(true)
{
OnFwd(OUT_AC, 75); Wait(3000);
OnRev(OUT_AC, 75); Wait(3000);
}
}
task main()
{
Precedes(music, movement);
}
```

8. Motores

```
task main()
{
    OnFwd(OUT_AC, 75);
    Wait(500);
    Off(OUT_AC);
    Wait(1000);
    OnFwd(OUT_AC, 75);
    Wait(500);
    Float(OUT_AC);
}
```