U.P.V. E.H.U.

Topic 3,

Robot programming

Outline

1. Lego Mindstorms NXT robot
2. NXT Central Processing Unit
3. Sensors
4. Engines
5. Other features
6. Programming
7. Development environments
8. Programming with NXC
9. Fundamentals of NXC

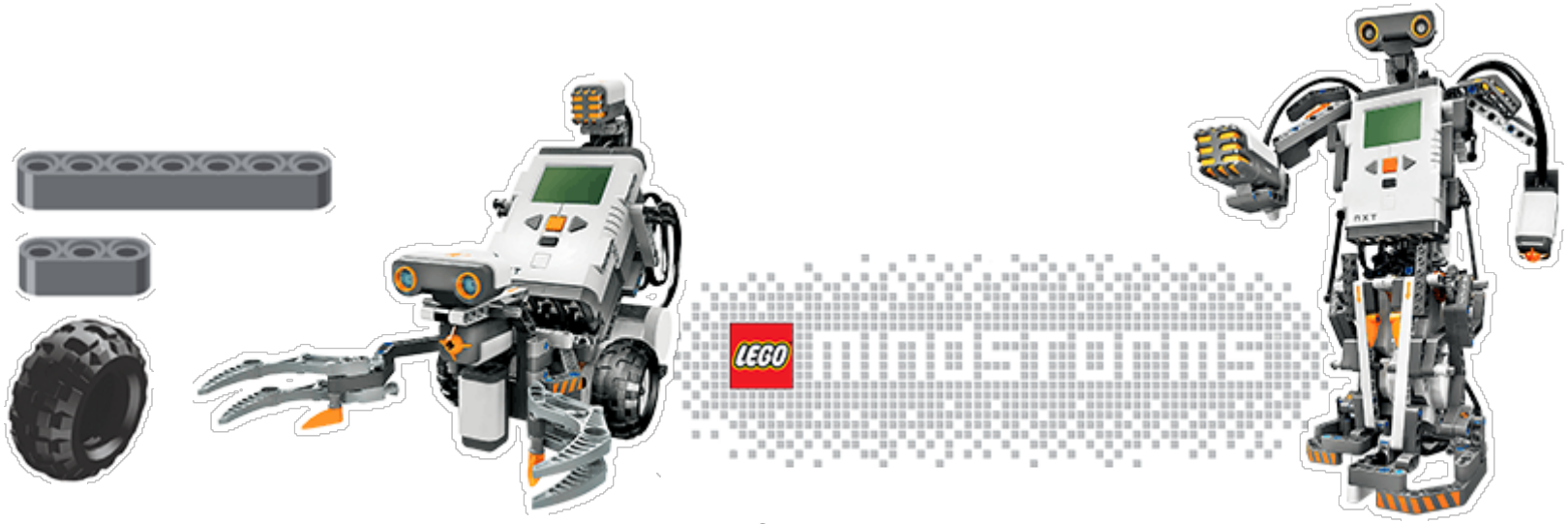
3.1:

Lego Mindstorms NXT robot

U.P.V. E.H.U.

Lego Mindstorms NXT robot

- It is a robot built by the Lego games brand.
- The advantage of this robot are that it can be assembled in various models, moreover if it is compared to others the robot package is economical, powerful and very versatile.



Advantages

- Easy to assemble and disassemble.
- Widely used (lots of information).
- Multiple mounting and programming possibilities.
- It is scalable, it can be expanded from the basics.
- It illustrates the main typical characteristics of embedded systems in real time.

Disadvantages

- Potentially weak structure.
- It is complicated to construct circular structures (rectangular pieces).
- The placement of batteries in the brick determines the shape.

3.2:

NXT Central Processing Unit

U.P.V. E.H.U.

NXT Central Processing Unit

- The central processing unit (brick) contains:
  - Monochrome LCD screen of 100x64 pixels.
  - 4 inputs (sensor reading).
  - 3 outputs (for motors).
  - Four control buttons (menus).
  - USB and Bluetooth interfaces.
  - Small integrated speaker.

U.P.V. E.H.U.

NXT Central Processing Unit

4 entradas

SENSOR_1 o S1

SENSOR_2 o S2

SENSOR_3 o S3

SENSOR_4 o S4

U.P.V. E.H.U.

NXT Central Processing Unit

3 output (motors)

OUT_A

OUT_B

OUT_C

U.P.V. E.H.U.

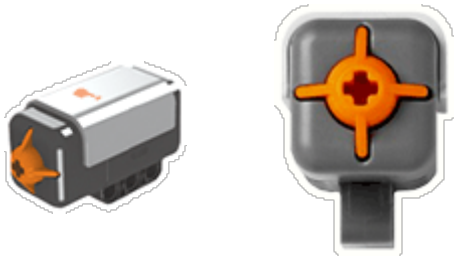
NXT Central Processing Unit

3.3:

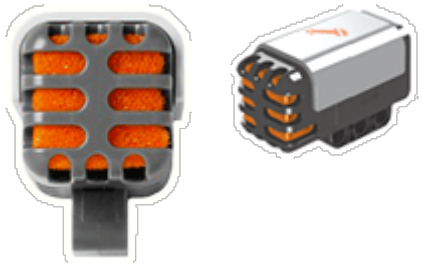
Sensors

Sensors

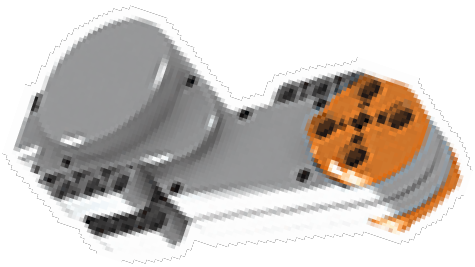
Light



Touch



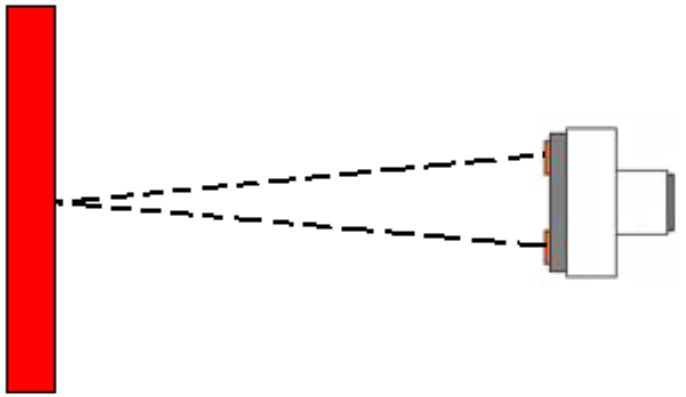
Sound



Rotation sensor

Sensors

Ultrasound: it works like a sonar, it sends an ultrasound waves front so that it reflects in the objects and it measures the time that takes in its echo, to know how far they are.

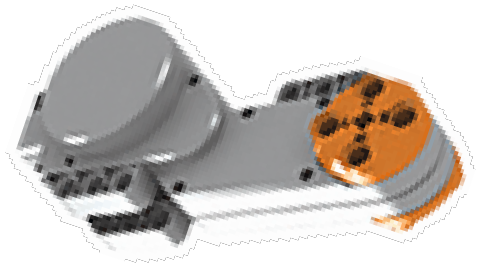


3.4:

Motors

Motors

- Rotation counter. The motor has a built-in rotation sensor, which allows for more precise turns.
- 360 degree rotation.
- The positions are relative.
- The maximum speed is 200 rpm.



3.5:

Other features

Other features

- It uses 6 AA batteries.
- It uses telephone-like (flattened) RJ-12 cables.
- Some manufacturers make other types of parts and sensors.

3.6:

Programming

Programming Tools

- NXT Software
- LabVIEW NEXT-G
- Microsoft Robotics Studio
- NXC (and NBC)
- Other:
  - Robot C
  - PbLua
  - Java

Programming

- Following, the steps to program an application for the robot:
- (1) The robot model is assembled (the steps indicated in the guide can be followed or developed on its own).
- (2) Programming is performed.

Programación

(3) Se descarga el programa en el Robot.

(4) Se ejecuta el programa en el Robot y se ven los resultados obtenidos.

(5) Importante: No se cuenta con un emulador para probar los desarrollos.

3.7:

Development environment

U.P.V. E.H.U.

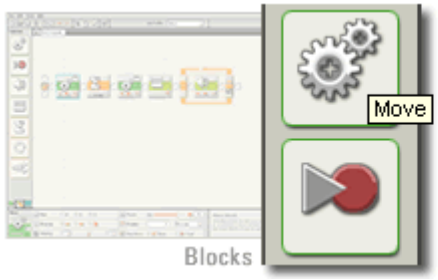
NXT development environment

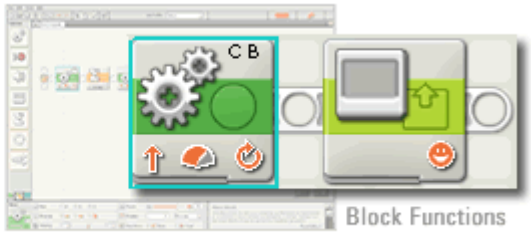
- Based on a graphical development environment (block programming), from National Instruments, LabView developers.
- Enables fast application execution. Useful for children.
- It is very limited for more complex programs, so other environments are used.

U.P.V. E.H.U.

NXT development environment

Building

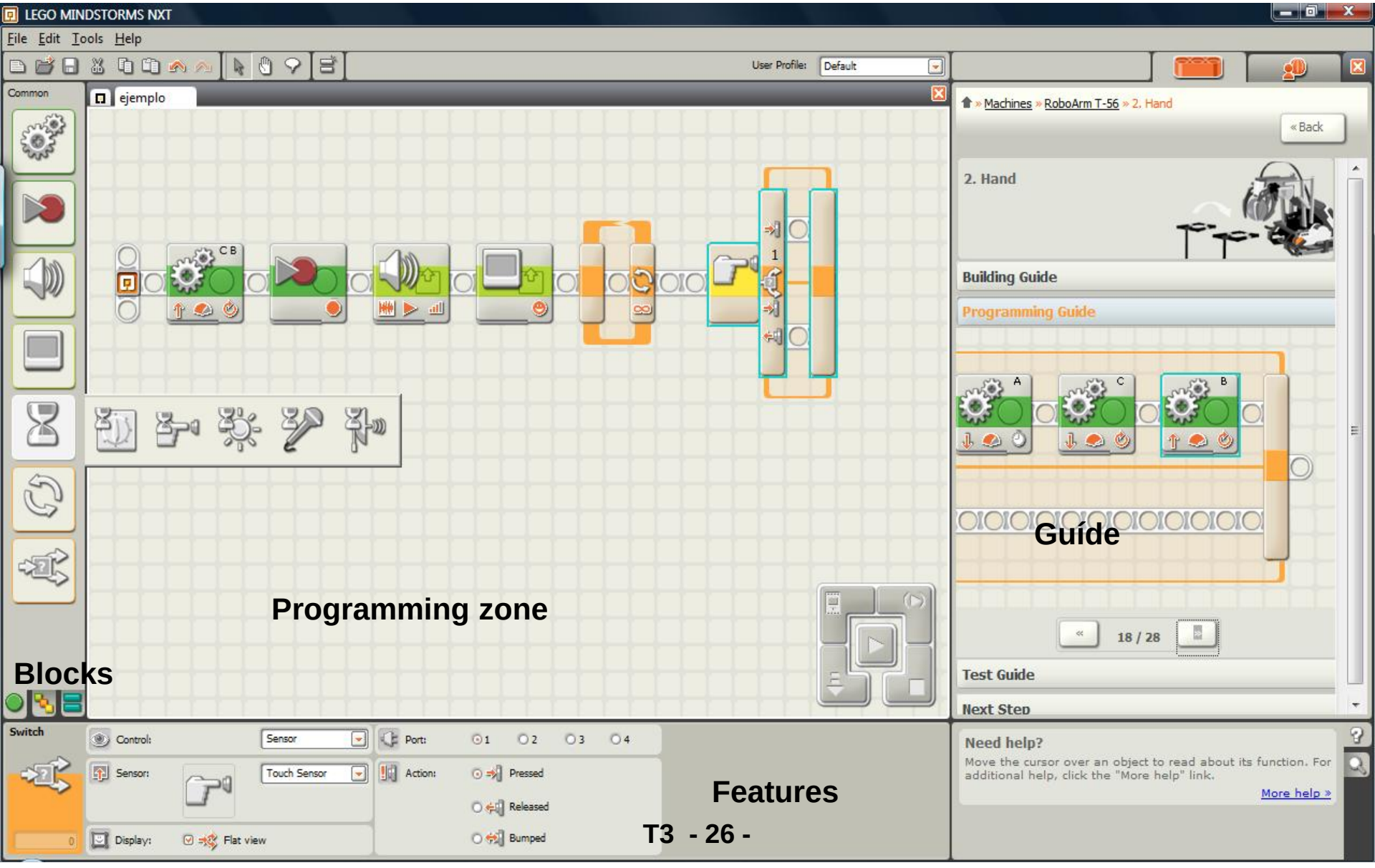




Programming

Running

NXT development environment



Other environments

- NXC (Not eXactly C) is a language similar to C, and is the most popular way to program the NXT. It is based on NBC (Next Byte Code).

- RobotC is a non-free environment.

3.8:

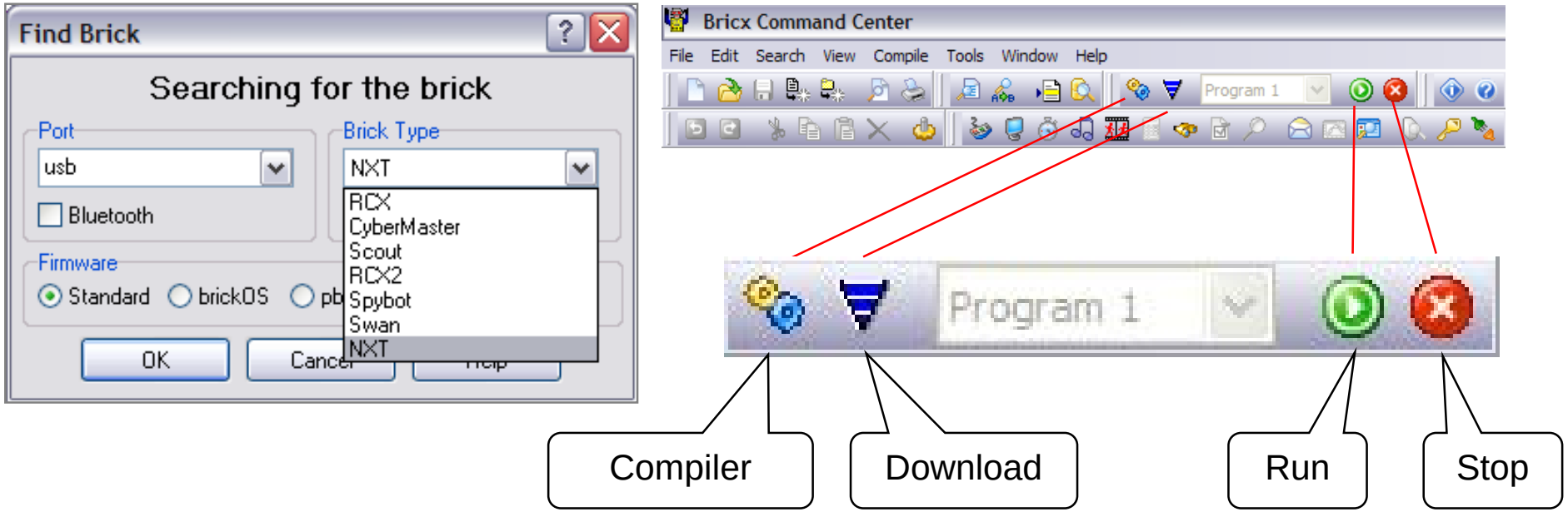
Programming by NXC

Programming by NXC

- NXC (Not eXactly C)
- There are some limitations such as:
  - Limited Memory
  - Limited to 256 tasks (task)

Programming by NXC

•BricxCC is a programming environment that allows to compile and load a program to the lego NXT:



3.9:

Fundamentals of NXC

NXC Program

- A program is composed of:
  - Tasks
  - Functions
  - Subroutines

U.P.V. E.H.U.

Reserved words

- NXC has reserved words that can not be used to name variables / functions / tasks ...

```
__RETURN__      char        long        sub
__RETVAL__      const       mutex       switch
__STRRETVAL__   continue    priority    task
__TMPBYTE__     default     repeat      true
__TMPWORD__     do          return      typedef
__TMPLONG__     else        safecall    unsigned
abs             false       short       until
asm             for         sign        void
bool            goto        start       while
break           if          stop
byte            inline      string
case            int         struct
```

#include

- #include works the same as in C, but there is no notion of path, so you can not use <>
- #include "NXCDefs.h" // In the current versions you do not need to include it manually
- #include <file.h> // ERROR !!!

Constants definition

#define name value

It will change name by its value. This can simplify changes when there is a constant that is used repeatedly.

Example:

#define waiting time 5000 // 5 seconds

U.P.V. E.H.U.

Variable statement

```
tipo nombre_variable ;  //variable sin valor inicial
tipo nombre_variable = valor ;   //variable con valor inicial
```

Ejemplos:

```
int x;              // x de tipo entero sin valor inicial
int y = 2;          // y de tipo entero con valor inicial igual a 2
char ch = 'a'; // ch de tipo char con valor inicial 'a'
```

U.P.V. E.H.U.

Task

```
Task name ()
{
// Task code
}
```

U.P.V. E.H.U.

Multi thread

- NXC supports multithreading, then it can perform several processes in parallel. A thread is created with the reserved word task.
- Every program has to have a main task (main) where the program starts.

```
1 task main()
2 {
3  /* esto es un comentario de
4  varias lineas
5  …..
6    y finaliza aquí */
7
8  //esto es un comentario de una linea
9 }
```

Functions

- A function groups a set of instructions/commands, and can be called when necessary.

- Return_function function_name (list_arguments)
- {
- // function body: set of instructions

Functions

- Time functions
  - Wait (time); // 1000 = 1 second
  - X = CurrentTick (); // current value of the internal timer in milliseconds
- Number Functions
  - X = Random (n); // random number between 0 and n-1
- LCD Functions
  - TextOut (x, y, message);
  - NumOut (x, y, value);
- Sound Features
  - PlayTone (frequency, duration);

U.P.V. E.H.U.

Simple program (1)

- Move the motors forward and backward:

  OnFwd (port, speed);
  OnRev (port, speed);
  Off (port);

Example:

```
Task main ()
{
    OnFwd (OUT_A, 75); // motor A forward at 75% of maximum speed
    OnFwd (OUT_C, 75); // motor C forward to 75% of maximum speed
    Wait (4000); // Wait 4 seconds
    OnRev (OUT_AC, 75); // engines A and C back to 75% of the maximum
    speed
    Wait (4000); // Wait 4 seconds
    Off (OUT_AC); // Stop the motors
}
```

U.P.V. E.H.U.

Simple program (1)

- Turn the robot by reversing the direction of rotation of one of the motors:

```
407/5000
#define TIME_MOVEMENT 800
#define 360_Time

Task main ()
{
OnFwd (OUT_AC, 75); // engines A and C forward to 75% of maximum speed
Wait (TIME_MOVEMENT); // Wait 0.8 seconds
OnRev (OUT_C, 75); // engine C backward at 75% of maximum speed
Wait (Wait_Time); // Wait 0.36 seconds
Off (OUT_AC); // Stop the motors
}
```

U.P.V. E.H.U.

Repeat commands

- repeat():

```
#define TIME_MOVEMENT 500
#define 000_Time 500

Task main () {
      Repeat (4) {
      OnFwd (OUT_AC, 75);
      Wait (TIME_MOVEMENT);
      OnRev (OUT_C, 75);
      Wait (TWAIN_TIME);
}
      Off (OUT_AC);
}
```

U.P.V. E.H.U.

Sensors

- Sensors:

SetSensor(*puerto, tipo***);**

SetSensorLight(*puerto***);** // light
SetSensorSound(*puerto***);** // sound
SetSensorTouch(*puerto***);** // contact
SetSensorLowspeed(*puerto***);** // ultrasound

U.P.V. E.H.U.

Sensors

- Take information from sensors:

  x = **Sensor(**IN_*n*)**;**

  x = **SensorUS(**IN_*n*)**;**

  SENSOR_1

  …

U.P.V. E.H.U.

Touch sensor

- Robot that advances until it collides with something:

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

Light sensor

- The light sensor can be configured to emit light or not, so that we can measure reflected light or ambient light in a certain direction. Measuring reflected light is especially useful to make a robot follow a line on the ground:

```
#define UMBRAL 40 //depende de las condiciones

Task main () {
    SetSensorLight (IN_3);
    OnFwd (OUT_AC, 75);
    While (true) {
        If (Sensor (IN_3)> THRESHOLD) {// off-track
            OnRev (OUT_C, 75);
            Wait (100); // softens movement
            Until (Sensor (IN_3) <= THRESHOLD); // on the track
            OnFwd (OUT_AC, 75);
        }
    }
}
```

U.P.V. E.H.U.

Sound sensor

- Program that waits for a loud sound to advance and stops when it detects another sound:

```
#define UMBRAL 40
#define MIC SENSOR_2
task main() {
    SetSensorSound(IN_2);
    while(true) {
        until(MIC > UMBRAL); //loud sound
        OnFwd(OUT_AC, 75);
        Wait(300);//Not to detect the same sound
        until(MIC > UMBRAL);
        Off(OUT_AC);
        Wait(300);
    }
}
```

Ultrasound sensor

- Robot dodging an obstacle before hitting it:

```
#define NEAR 15 //cm
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);
    }
}
```