Fundamentos de Robótica

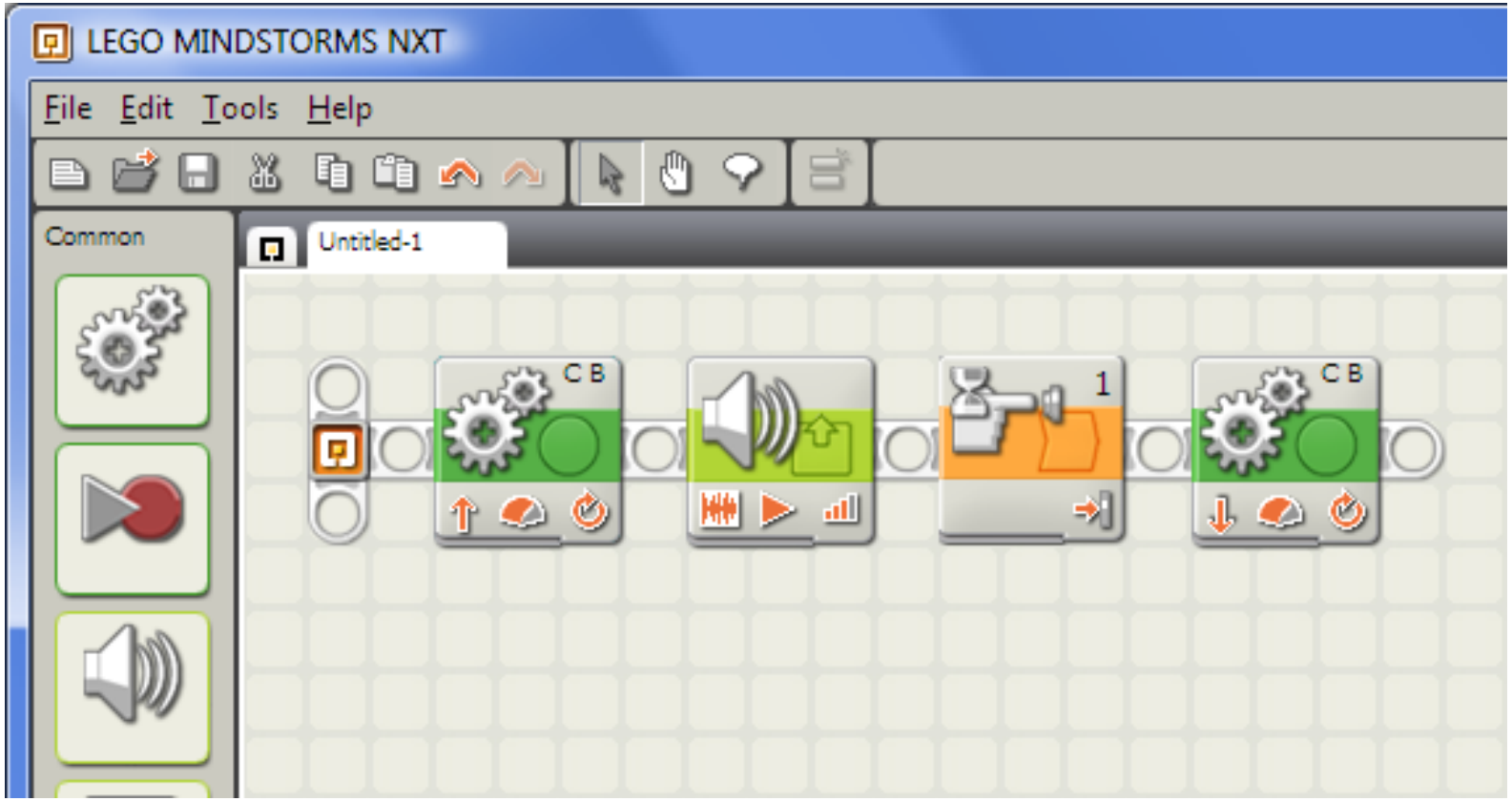
BricxCC y NXC

Departamento de Ingeniería en Sistemas y Computación
Universidad Católica del Norte, Antofagasta.

Lego Mindstorms NXT

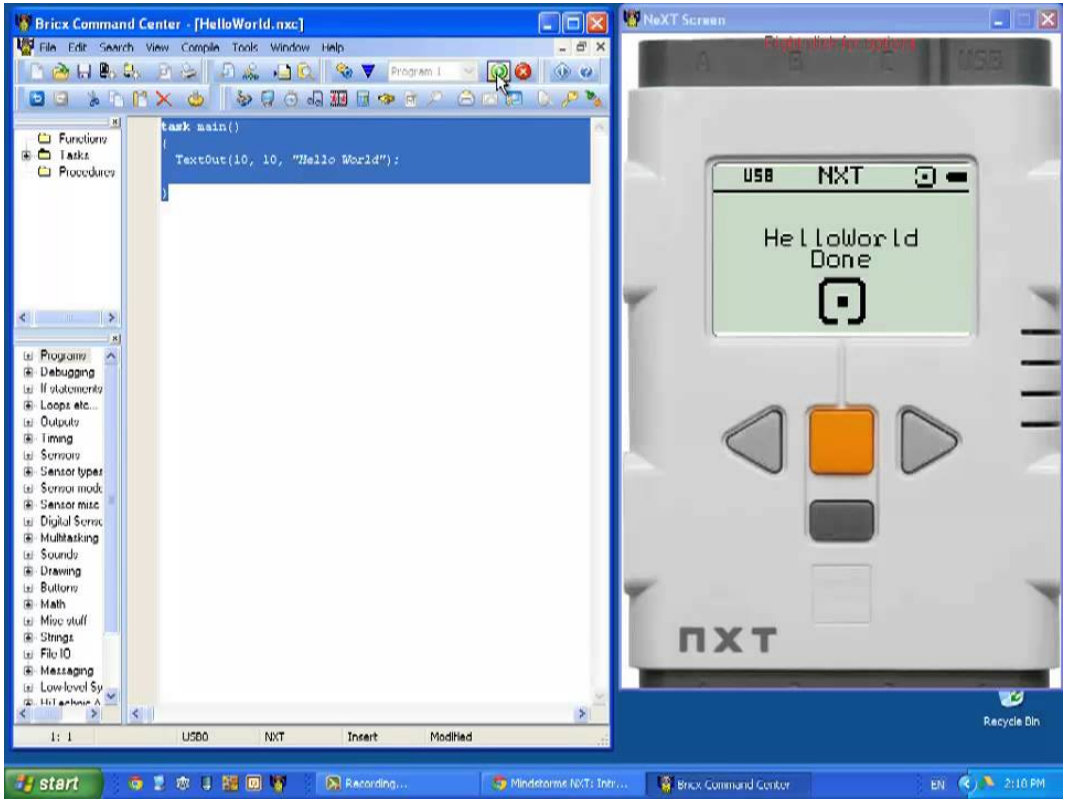
- Ladrillo (Brick, contiene microcontrolador)
- 3 motores
- 4 sensores
  - Sensor de tacto
  - Sensor de luz
  - Sensor de ruido
  - Sensor ultrasónico

NXT-G



- Entorno de programación en bloques
- Estructuras condicionales y cíclicas también se manejan por bloques
- Se pueden definir rutinas *paralelas*

BricxCC



- Entorno de desarrollo con instrucciones de alto nivel para manejar componentes de Lego Mindstorms
- Lenguaje utilizado se denomina NXC (Not eXactly C)
- Rutina principal asociada a **task main()**
- Existen otras alternativas de pago (como RobotC)

Programas de ejemplo extraídos de literatura sugerida:

Variables pre-definidas

- `OUT_A`, `OUT_AB`
- `SENSOR_1`
- para declarar un sensor de tacto en la entrada 1:
  `SetSensor(IN_1, SENSOR_TOUCH)` o
  `SetSensorTouch(IN_1)`

Instrucciones relevantes

- OnFwd(Motor, potencia): por ejemplo `OnFwd(OUT_A, 50)`, o en sentido contrario `OnFwd(OUT_A, -50)`. Notar que pueden activarse dos motores al mismo tiempo con `OnFwd(OUT_AB, 50)`
- Wait(milisegundos): por ejemplo `Wait(2000)` ejecuta una espera de `2` segundos
- OnRev(motor, potencia): equivalente a `OnFwd(motor, -potencia)`
- Off(motor): apaga el motor, por ejemplo `Off(OUT_A)` o también `Off(OUT_AB)`

Estructuras de condición y repetición

- disponibles `if-else`, `while`, `for` tal como en `C`
- `repeat(n){}` repite *n* veces las instrucciones que se encuentren dentro de los paréntesis de llave
- `until(expresión)` realiza una espera hasta que `expresión` es `verdadero`

Programa de ejemplo

```
#define MOVE_TIME   1000
#define TURN_TIME    500

task main()
{
  repeat(10)
  {
    repeat(4)
    {
      OnFwd(OUT_AC, 75);
      Wait(MOVE_TIME);
      OnRev(OUT_C, 75);
      Wait(TURN_TIME);
    }
  }
  Off(OUT_AC);
}
```

- Para ejecutar el programa, pero se debe *compilar* y luego *descargar*.

Subrutinas y Tareas

- para llamar a una subrutina previamente definida, basta con referenciarla por su nombre
- para iniciar una tarea (ejecutada de forma concurrente) se utiliza **StartTask(nombre_tarea)**

```
sub turn_around(int pwr)
{
  OnRev(OUT_C, pwr); Wait(900);
  OnFwd(OUT_AC, pwr);
}

task main()
{
  OnFwd(OUT_AC, 75);
  Wait(1000);
  turn_around(75);
  Wait(2000);
  turn_around(75);
  Wait(1000);
  turn_around(75);
  Off(OUT_AC);
}
```

Otras funciones útiles

Mostrar mensaje (string) en linea `X`, segmento (horizontal) `n_segmento`:

- TextOut(n_segmento, LCD_LINE_X, mensaje)

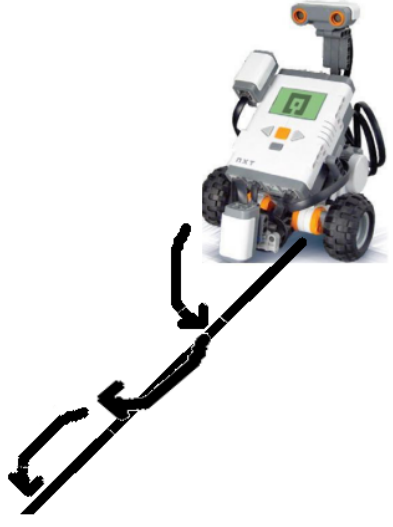
Emitir tono de cierta `frecuencia` y `duración`:

- PlayTone(frecuencia, duración)

Seguidor de líneas

Estrategia básica para seguir una línea negra con 1 sensor:

- identificar rango de valores que arroja el sensor de luz para `blanco` (color de la superficie) y `negro` (color de la línea a seguir)
- programar rutina de forma que al detectar `negro`, gire a la `derecha` y al detectar `blanco`, gire a la `izquierda`
- el comportamiento formado se ilustra en la imagen:



- los giros se deben realizar tomando la **otra rueda como pivote**, no sobre su propio eje.

Tipos de giros

- giro a la izquierda sobre su propio eje:
  `OnFwd(OUT_A, 50); OnFwd(OUT_B, -50);`
- giro a la izquierda con rueda B como pivote:
  `OnFwd(OUT_A, 50); Off(OUT_B);`

Sensor de luz

alternativas (suponiendo conexión a entrada 3:

- SetSensorLight(IN_3)
- SetSensorType(IN_3,IN_TYPE_LIGHT_ACTIVE);
- SetSensorType(IN_3,IN_TYPE_LIGHT_INACTIVE);
- SetSensorMode(IN_3,IN_MODE_PCTFULLSCALE)

Actividad

- Según el procedimiento anteriormente descrito, programe una rutina en su **plataforma armada**, para que sea capaz de seguir una línea negra **por la derecha** con **sólo 1 sensor** de luz.