Lego EV3 Programming Software

Georgia Institute of Technology
ICE Programs
Mr. Michaud
User Interface Windows

Program Tabs

Programming Area: Where the blocks are assembled to create program.

Active code tabs within program

Block selection Window

Brick Communication Window
Flow Control Blocks

Start
Wait
Loop (Do Until)
Switch (If Else)
Loop Interrupt
Raw Sensor Data Blocks

These blocks provide data output of the value of the indicated sensor.
Data Operations Blocks

These blocks control variables, logic, math, random, text, and range operations.

To create a variable, drag a variable block to the work area and select “Add Variable”
Advanced Blocks

Blocks for file access, communication, bluetooth, raw data, and advanced motor functions.
MyBlocks

These are blocks that you create. Similar to functions or procedures.
Brick Status Window

- **Selection Tabs**
- **Brick Info Mode**
- **Port Info Mode**
- **Available Bricks**

- **Download Program**
- **Download and play Program**
- **Download and play selected blocks**
Single Motor Block Parameters

Motor Block Mode Selection:
- Seconds
- Degrees
- Rotations

Motor Block Power Selection
Single Motor Block Parameters

Select Number of Rotations (Degrees, Time)

Select Brake or Coast Mode
Single Motor Block Parameters

Select Motor Port
Dual Motor Blocks

Tank Drive Settings (Motor power for two motors set separately.)

Move Steering Settings. (Motors set according to arc (steering left and right).

Dual motor blocks use PID steering algorithms to keep two drive motors in sync during program.
Program Samples: Forward for Degrees

```c
void steeringMotors(int B, int C, int start, int end, int degrees, int action);
```

```c
int main()
{
    steeringMotors(B, C, 0, 75, 360, brake);
}
```
task main() {
    steeringMotors(B, C, 0, 50);
    while (SensorValue[touch] == 0) {
    }
    steeringMotors(B, C, 0, 0, brake);
}
Program Samples: Forward and Wait for Touch Separate Motors

task main() {
    motor[B]=50;
    motor[C]=50;
    while (SensorValue[touch] == 0) {
    
    }
    motor[B] = 0;
    motor[C] = 0;
}
Program Samples: Forward until Sonar and Turn

- Motors B + C On for 50 power
- Wait until Sonar is less than 20 centimeters
- Motors B + C to off.
- Wait for 0.25 seconds
- Point turn right at 30 power with wheels turning 180 degrees.
Programs: Forward and Turn with loop

- Motors B + C On for 50 power
- Wait until Sonar is less than 20 centimeters
- Motors B + C to off.
- Wait for 0.25 seconds
- Point turn right at 30 power with wheels turning 180 degrees.

Infinity symbol at end of loop means infinite loop.
Other Types of Loops

- Motors B + C On for 50 power
- Wait until Sonar is less than 20 centimeters
- Motors B + C to off.
- Wait for 0.25 seconds
- Point turn right at 30 power with wheels turning 180 degrees.
Example with Loop and Logic Blocks

- **Motors B + C On for 50 power.**
- **Loop until Sonar is less than 20 or touch sensor is triggered.**
- **If Sonar is less than 30 centimeters Wait 0.25 seconds Point turn Right.**
- **Wait for 0.25 seconds.**

- **Motors B + C to off.**
Loop within Loop
Objectives

- Linear Programming: Maze Navigation
- Sensor / Event Driven Programming
  - Touch Stop
  - Sonic Stop
  - Line Following
    - Heaveside
    - Proportional
Line Follow Algorithms

- Heaviside

\[ H[n] = \begin{cases} 
0, & n < 0, \\
1, & n \geq 0, 
\end{cases} \]

- Proportional

```plaintext
float kP = 0.5;
int threshold = 40;
int lightValue = 0;
int error = 0;
int power = 50;
int turn = 0;
int leftP = 0;
int rightP = 0;

while (true) {
    error = lightValue - threshold;
    turn = error * kP;
    leftP = power + turn;
    rightP = power - turn;
    motor[right] = rightP;
    motor[left] = leftP;
}
```
Artbotics