THYMIODSL

A COMPREHENSIVE GUIDE TO PROGRAM THYMIO ROBOTS USING THYMIODSL

WRITTEN BY:
- LUÍS SANTOS
- MIGUEL CUT

TABLE OF CONTENTS

1. Introduction

Welcome to the ThymioDSL Tutorial!

This comprehensive guide is designed to help you learn how to program the Thymio robots using ThymioDSL, a **domain-specific language (DSL)** created specifically for this purpose. ThymioDSL offers an intuitive and engaging way to control the Thymio robots, making it a great tool for an introduction to programming and robotics.

What is Thymio?

Thymio is an educational robot developed to teach students about robotics and programming. Equipped with various sensors and capabilities, Thymio robots can interact with its environment in numerous ways. It's an ideal platform for learning because it combines hands-on experience with programming experience, providing immediate feedback as users see their programs come to life.

What is ThymioDSL?

ThymioDSL is a specialized programming language designed by Luís Santos and Miguel Cut, students at Faculty of Sciences of the University of Lisbon, with the objective of simplifying the process of coding for Thymio robots. Built using the Xtext framework, ThymioDSL provides a user-friendly syntax that generates Aseba code, the underlying language that Thymio robots understand. This DSL focuses on making programming accessible to users with no prior experience, using clear and descriptive commands.

Goals of this Tutorial:

- To show the components and features of Thymio robots;
- Show how to setup and connect Thymio to your computer;
- To introduce the basics of ThymioDSL, including its syntax and structure;
- Explore simple programming commands to control the robot's movements and sensors;
- Implement some basic cases, like line following and obstacle avoidance.

Why learn Robotics and Programming?

Learning robotics and programming at an early age has numerous benefits. It enhances problem-solving skills, fosters creativity, and promotes critical thinking. Additionally, as a student who may follow a carrer in this area, you might want to explore and have a taste of what programming and robotics are, and this tutorial is the best example to begin with.

How to use this tutorial?

This tutorial is structured to guide you step-by-step from the functioning of the robot to the basics of the ThymioDSL. Each section builds on the previous one, ensuring a gradual and comprehensive learning experience. You'll start with an introduction to Thymio's hardware and capabilities, move on to setting up your programming environment, and then dive into writing your first programs using ThymioDSL. Let's get started!

2.Getting to know the Thymio Robot

The Thymio robot is a versatile educational tool designed to introduce users to the fundamentals of robotics and programming. In this section, we will explore the main components and features of the Thymio robot, in order to proceed with ThymioDSL.

Key Features and Components

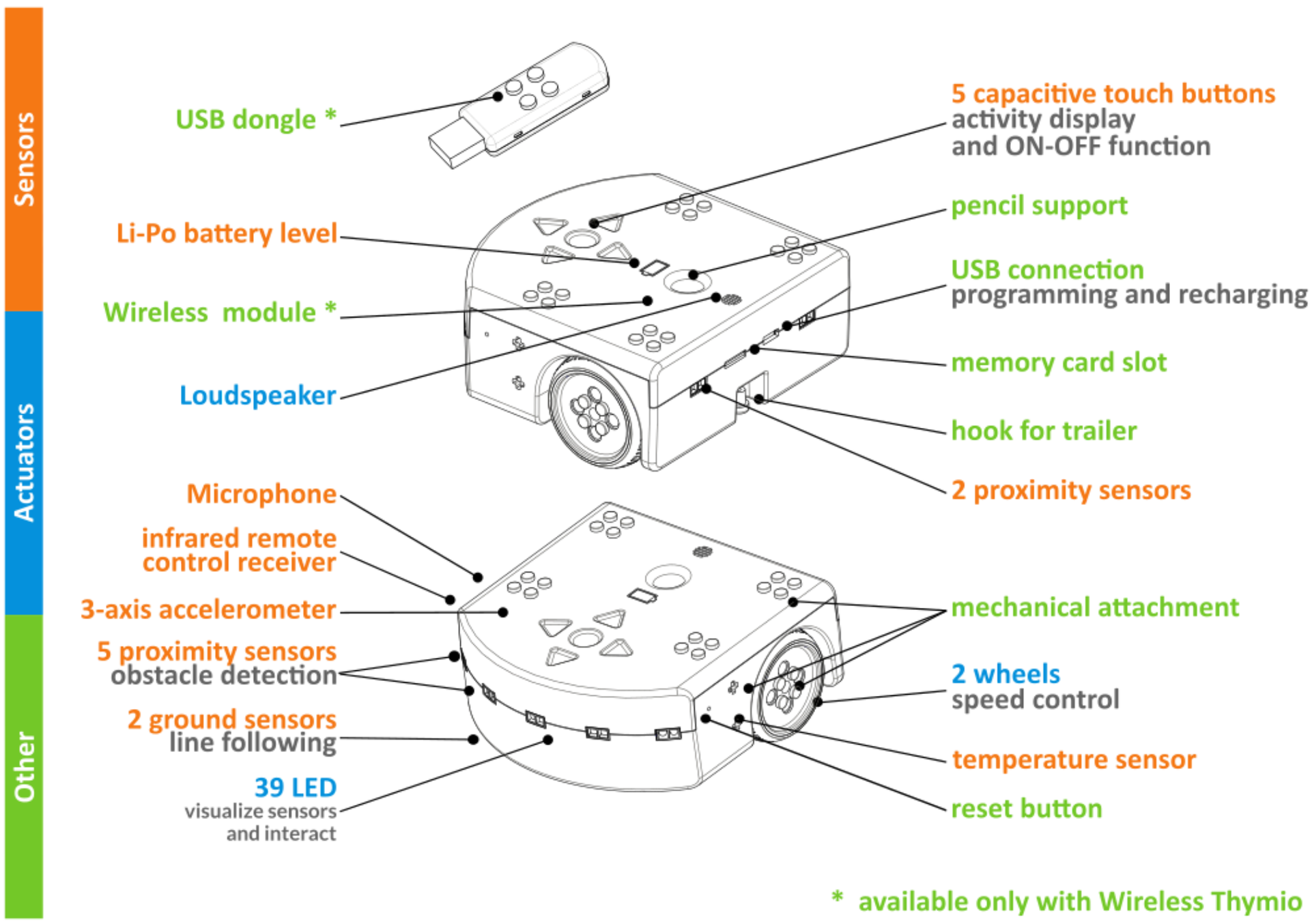
- **USB Pen Connection:** Thymio connects to your computer via a USB pen, which is used for programming;
- **Battery level display and charger connection:** The robot displays its battery level, ensuring you know when it needs recharging. You can recharge it with the charger that's in the robot's box;
- **Loudspeaker:** The robot has a built-in loudspeaker, which can be used to play sounds, enhancing the interactivity of your programs;
- **Capacitive Touch Buttons:** Thymio is equipped with five capacitive touch buttons. These buttons are used for controlling the robot and include an activity display and ON-OFF function. They provide a simple way to interact with the robot without needing a computer;
- **Proximity Sensors:** Thymio has 5 proximity sensors in the front and sides to detect obstacles.
- **Microphone:** Thymio has a built-in microphone, allowing it to detect and respond to sounds. This feature can be used in sound-based interaction programs.
- **Ground Sensors:** The robot has two ground sensors that can be used for line-following tasks. These sensors can detect the contrast between black and white surfaces, allowing Thymio to follow a line on the ground.
- **LEDs for Visualization and Interaction:** Thymio is equipped with various LEDs to visualize sensor data and interact with users. These LEDs can indicate the robot's status, sensor readings, and provide feedback while they're executing a program.
- **Wheels and Speed Control:** Thymio has two wheels that provide mobility and speed control. These wheels are independently controlled, enabling precise movement and turning.

Connecting Thymio to your computer

To begin programming Thymio, you will need to connect it to your computer using the provided USB pen. Follow these steps:

- **Install necessary Software:** Ensure that you have installed the required software to program with Thymio, such as ThymioSuite.
- **Power On Thymio:** Press the ON-OFF button to turn on Thymio;
- **Check Battery Level:** Ensure Thymio has sufficient battery life for your programming session. The battery level is displayed on the robot.
- **Connect via USB:** Insert the USB pen into the USB port of your computer to connect;

Below you can see a basic diagram where it shows the location of buttons and capabilities of Thymio robots.



By understanding and using these features, you will be well-prepared to start programming with Thymio and ThymioDSL and explore the endless possibilities it offers for learning and experimentation.

3. Basics of ThymioDSL

In this section, we will go into the basics of ThymioDSL, an intuitive language designed to simplify the programming of Thymio robots. We will cover the syntax of ThymioDSL, how to structure a program, and the various commands and inputs that can be used.

Syntax of ThymioDSL Programs

The programs written in ThymioDSL language are basically composed by **Procedures.** Each **Procedure** has a unique name, and they are composed by **1 Event** and **multiple Actions**. So, in few words, we can say that a Procedure defines a trigger (Event), and when something is triggered (Event), the Thymio robot will act as you programmed it in Actions, which means it does all the actions you wrote in your code. It's like giving orders to the robot, when something happens.

This structure mirrors the Visual Programming Language (VPL) representation used in Thymio Suite Software, where each block represents a Procedure.

Now speaking about **Events**, in ThymioDSL Events can be one of **4 types**:

- **button_is_clicked**, which receives one of the four orthogonal directions (up,down,left,right) plus the center as input, corresponding to the robot's buttons;
- **robot_detects_stimulus**, which receives either "tap" or "sound" as input;
- **proximity_sensor_is_activated**, which receives none or any combination of the seven horizontal proximity sensors that the robot has (front_left, front_center_left, front_center, front_center_right, front_right, back_left, back_right). Sensors can be set as "far" (no selection/gray in VPL), "close" (black in VPL), and "very_close" (white in VPL). Omission sets the value as "far"; **Note: The order of sensors must be respected**. If the user starts with the front_center_left sensor and omits the front_left sensor, the front_left sensor will be set to "far" and cannot be defined after the front_center_left, only before.

- **bottom_sensor_detects_color**, which receives inputs from the two bottom sensors (left, right), which can be set as “any” (no selection/gray in VPL), “white” (white in VPL), or “black” (black in VPL);

Now speaking about Actions, in ThymioDSL Actions can be of 3 types:

- **move**, which corresponds to the activation of the left and right motors, which receive Expressions as input. The acceleration values must range between -500 and 500;
- **lights**, which corresponds to the robot’s top and bottom lights. These can be set to on or off. If set to on, they receive an RGB expression, where each value is an Expression;
- **sound**, which receives up to six notes, each with a pitch and duration. The pitch is an Expression, and the duration can be either “short” (black in VPL) or “long” (white in VPL).

Now, what about Expressions?

Expressions in ThymioDSL are used to define the inputs for Actions. They are a fundamental part of the language, allowing for calculations and logic to be implemented within the robot’s behavior. Expressions are similar to expressions in mathematics, this is, they can be numbers, sum of numbers, multiplication of numbers or even division.

4. Simple Use Cases

In this section, we will explore some practical examples of programming a Thymio robot using ThymioDSL. This example demonstrates how to create a set of procedures that enable the robot to follow a line, make turns based on sensor input, and stop when a button is pressed.

Example 1: Line Following with Turns

This example includes five procedures: steer_right, steer_left, follow_line, u_turn, and stop. Each procedure defines specific events and corresponding actions to control the robot's behavior.

```
Procedure: steer_right
     Event:
          bottom_sensor_detects_color:
               left: white
               right: black
     Actions:
          lights:
               top_light: on (16*(20/(5+5)),0,0)
               bottom_light: on (0,0,16*(20/(5+5)))
          move:
               left_motor: 500
               right_motor: 0
```

This is what a procedure would look like in ThymioDSL. Some things you can observe in the program are that the top light turns on with a red color, the bottom light turns on with a blue color, the left motor runs at full speed and the right motor is set to stop. All of this actions occur when there is a trigger (Event)

This procedure steers the robot to the right when the left bottom sensor detects white and the right bottom sensor detects black.

```
Procedure: steer_left
     Event:
          bottom_sensor_detects_color:
               left: black
               right: white
     Actions:
          lights:
               top_light: on (0,0,16*(20/(5+5)))
               bottom_light: on (16*(20/(5+5)),0,0)
          move:
               left_motor: 0
               right_motor: 500
```

This procedure steers the robot to the left when the left bottom sensor detects black and the right bottom sensor detects white. Some things you can observe in the program are that the top light turns on with a blue color, the bottom light turns on with a red color, the left motor stops and the right motor runs at full speed.

```
Procedure: follow_line
     Event:
          bottom_sensor_detects_color:
               left: black
               right: black
     Actions:
          lights:
               top_light: on (0,16*(20/(5+5)),0)
               bottom_light: on (0,16*(20/(5+5)),0)
          move:
               left_motor: 500
               right_motor: 500
```

This procedure makes the robot follow the line when both bottom sensors detect black. Some things you can observe in the program are that the top and bottom lights turn on with a green color, and the left and right motors run at full speed.

```
Procedure: u_turn
     Event:
          bottom_sensor_detects_color:
               left: white
               right: white
     Actions:
          move:
               left_motor: 500
               right_motor: 0
```

This procedure initiates a U-turn when both bottom sensors detect white. Some things you can notice in this program are that the left motor runs at full speed and that the right motor stops.

```
Procedure: stop
     Event:
          button_is_clicked:
               center
     Actions:
          move:
               left_motor: 641798 * 0
               right_motor: 641798 * 0
```

This procedure stops the robot when the center button is clicked. Some things you can notice in this program are that the left and right motors stop (because of the 0).

And this is what the line following program would look like.

Try out this program on your Thymio robot to see the results!

Example 2: Obstacle avoidance

This example includes six procedures: detect_left_wall, detect_near_left_wall, detect_right_wall, detect_near_right_wall, no_obstacles, and stop. Each procedure defines specific events and corresponding actions to control the robot's behavior.

```
Procedure: detect_left_wall
     Event:
          proximity_sensor_is_activated:
               front_left: close
               front_center_left: close
               front_center: close
Actions:
     move:
          left_motor: 500
          right_motor: 0
     lights:
          top_light: on (0, 0, 32)
          bottom_light: on (0, 0, 32)
```

This procedure steers the robot to the right when the left proximity sensors detect a close obstacle. Some things you can notice in this program are that the left motor runs at full speed, the right motor stops, and the top and bottom lights turn on with a blue color.

```
Procedure: detect_near_left_wall
     Event:
          proximity_sensor_is_activated:
               front_left: very_close
               front_center_left: very_close
               front_center: very_close
Actions:
     move:
          left_motor: 500
          right_motor: -500
     lights:
          top_light: on (32, 0, 0)
          bottom_light: on (32, 0, 0)
```

This procedure causes the robot to turn sharply to the right when the left proximity sensors detect a very close obstacle. Some things you can notice in this program are that the left motor runs at full speed, the right motor runs in reverse at full speed and that the top and bottom lights turn on with a red color.

```
Procedure: detect_right_wall
        Event:
            proximity_sensor_is_activated:
            front_center: close
            front_center_right: close
            front_right: close
    Actions:
            move:
                left_motor: 0
                right_motor: 500
            lights:
                top_light: on (0, 0, 32)
                bottom_light: on (0, 0, 32)
```

This procedure steers the robot to the left when the right proximity sensors detect a close obstacle. Some things you can notice in this program are that the left motor stops, the right motor runs at full speed, and the top and bottom lights turn on with a blue color.

```
Procedure: detect_near_right_wall
     Event:
          proximity_sensor_is_activated:
          front_center: very_close
          front_center_right: very_close
          front_right: very_close
     Actions:
          move:
               left_motor: -500
               right_motor: 500
          lights:
               top_light: on (32, 0, 0)
               bottom_light: on (32, 0, 0)
```

This procedure causes the robot to turn sharply to the left when the right proximity sensors detect a very close obstacle. Some things you can notice in this program are that the left motor runs in reverse at full speed, the right motor runs at full speed, and the top and bottom lights turn on with a red color.

```
Procedure: no_obstacles
     Event:
          proximity_sensor_is_activated:
          front_center: far
     Actions:
          move:
               left_motor: 250
               right_motor: 250
          lights:
               top_light: on (0, 32, 0)
               bottom_light: on (0, 32, 0)
```

This procedure moves the robot forward when the front center sensor detects no nearby obstacles. Some things you can notice in this program are that the left and right motors run at half speed, and the top and bottom lights turn on with a green color.

```
Procedure: stop
     Event:
          proximity_sensor_is_activated:
          back_left: close
          back_right: close
     Actions:
          move:
               left_motor: 0
               right_motor: 0
          lights:
               top_light: on (0, 0, 0)
               bottom_light: on (0, 0, 0)
```

This procedure stops the robot when the back sensors detect a close obstacle. Some things you can notice in this program are that the left and the right motors stop, and the top and bottom lights turn off.

And this is what the obstacle avoidance program would look like.

Try out this program on your Thymio robot to see the results!

Example 3: Sound and Tap Responses

This example includes two procedures: **clap_for_red** and **tap_for_music**. Each procedure defines specific events and corresponding actions to control the robot's behavior.

```
Procedure: clap_for_red
     Event:
          robot_detects_stimulus: sound
     Actions:
          lights:
               top_light: on (32, 0, 0)
               bottom_light: off
```

This procedure responds to the sound stimulus (e.g., a clap) by turning on the top light with a red color and turning off the bottom light. Some things you can notice in this program are that the top light turns on with a red color and the bottom light turns off.

```
Procedure: tap_for_music
     Event:
          robot_detects_stimulus: tap
     Actions:
          sound:
               note: 1 short
               note: 2 long
               note: 0 short
               note: 4 long
               note: 5 short
```

This procedure responds to the tap stimulus by playing a sequence of musical notes with varying durations. Some things you can notice in this program are that the robot plays the notes 0, 1 and 5 for a short duration, and 2 and 4 for long duration, but in the given order.

And this is what the sound and tap responses program would look like.

Try out this program on your Thymio robot to see the results!

5. Complex Use Case

In this section, we will explore a complex example of programming a Thymio robot using ThymioDSL. This example demonstrates various procedures that enable the robot to respond to different sensor inputs and user interactions, allowing it to follow lines, avoid obstacles, and perform specific actions upon detecting sounds or taps. Here is what the program would look like:

```
Procedure: steer_right
    Event:
        bottom_sensor_detects_color:
            left: white
            right: black
    Actions:
        lights:
            top_light: on (16*(20/(5+5)),0,0)
            bottom_light: on (0,0,16*(20/(5+5)))
        move:
            left_motor: 500
            right_motor: 0


Procedure: steer_left
    Event:
        bottom_sensor_detects_color:
            left: black
            right: white
    Actions:
        lights:
            top_light: on (0,0,16*(20/(5+5)))
            bottom_light: on (16*(20/(5+5)),0,0)
        move:
            left_motor: 0
            right_motor: 500
```

```
Procedure: follow_line
     Event:
          bottom_sensor_detects_color:
               left: black
               right: black
     Actions:
          lights:
               top_light: on (0,16*(20/(5+5)),0)
               bottom_light: on (0,16*(20/(5+5)),0)
          move:
               left_motor: 500
               right_motor: 500


Procedure: u_turn
     Event:
          bottom_sensor_detects_color:
               left: white
               right: white
     Actions:
          move:
               left_motor: 500
               right_motor: 0


Procedure: stop
Event:
     button_is_clicked: center
Actions:
     move:
          left_motor: 641798 * 0
          right_motor: 641798 * 0
```

```
Procedure: detect_left_wall
     Event:
          proximity_sensor_is_activated:
               front_left: close
               front_center_left: close
               front_center: close
     Actions:
          move:
               left_motor: 500
               right_motor: 0
          lights:
               top_light: on (0, 0, 32)
               bottom_light: on (0, 0, 32)
```

```
Procedure: detect_near_left_wall
     Event:
          proximity_sensor_is_activated:
               front_left: very_close
               front_center_left: very_close
               front_center: very_close
     Actions:
          move:
               left_motor: 500
               right_motor: -500
          lights:
               top_light: on (32, 0, 0)
               bottom_light: on (32, 0, 0)
```

```
Procedure: detect_right_wall
      Event:
            proximity_sensor_is_activated:
                  front_center: close
                  front_center_right: close
                  front_right: close
      Actions:
            move:
                  left_motor: 0
                  right_motor: 500
            lights:
                  top_light: on (0, 0, 32)
                  bottom_light: on (0, 0, 32)


Procedure: detect_near_right_wall
      Event:
            proximity_sensor_is_activated:
                  front_center: very_close
                  front_center_right: very_close
                  front_right: very_close
      Actions:
            move:
                  left_motor: -500
                  right_motor: 500
            lights:
                  top_light: on (32, 0, 0)
                  bottom_light: on (32, 0, 0)


Procedure: force_right_turn
      Event:
            button_is_clicked: right
      Actions:
            move:
                  left_motor: 500
                  right_motor: -500
```

```
Procedure: force_left_turn
      Event:
            button_is_clicked: right
      Actions:
            move:
                  left_motor: -500
                  right_motor: 500

Procedure: force_reverse
      Event:
            button_is_clicked: down
      Actions:
            move:
                  left_motor: -250*2
                  right_motor: -100*5


Procedure: turn_lights_off
      Event:
            robot_detects_stimulus: tap
      Actions:
            lights:
                  top_light: off
                  bottom_light: off
Procedure: play_music
      Event:
            robot_detects_stimulus: sound
      Actions:
            sound:
                  note: 5 long
                  note: 2*2 short
                  note: 3000/(500+500) long
                  note: 0 long
                  note: 1+1 short
                  note: -1*-1 long
```

Try out this program on your Thymio robot to see the results!

6. Conclusion

In this tutorial, we covered the essential aspects of programming the Thymio robot using ThymioDSL. Here's a brief summary of the key concepts you learned:

- **Introduction to Thymio**: Understanding the various components and features of the Thymio robot, including its sensors, actuators, and connectivity options.
- **Setting Up**: How to connect Thymio to your computer, install the necessary software, and ensure the robot is ready for programming.
- **ThymioDSL Basics**: The structure of ThymioDSL programs, including Procedures, Events, and Actions.
- **Practical Examples**: Step-by-step guides on creating programs for various robot behaviors such as line following and obstacle avoidance.

Encouragement to Explore Further

With the foundational knowledge you've gained, you are now equipped to start experimenting with more complex programs and exploring the full potential of ThymioDSL. Robotics and programming offer endless possibilities for creativity and problem-solving. Don't hesitate to challenge yourself with new programs and ideas.

Advanced Topics and Next Steps

As you become more comfortable with ThymioDSL, consider exploring advanced topics such as combining multiple sensors and actions to create intricate robot behaviors, like maze solving or leader-follower scenarios.

Remember, the journey of learning robotics and programming is continuous. Keep experimenting, learning, and having fun with your Thymio robot!

7. Additional Resources

To support your ongoing learning and exploration, here are some valuable resources:

- Thymio Teaching Materials
- VPL3 Documentation