Turtlebot Project

For the final lab assignment, you'll be developing a statechart model to make a Turtlebot explore and map a maze. You'll create this model using the ItemisCREATE statechart tool and run it on a Turtlebot3 using ROS2, the robot operating system.

Form teams of four for this assignment, dividing tasks among members for efficiency. Suggested tasks include manual control, the maze exploration algorithm, wall detection, calibration, and grid updates.

A benefit of the Turtlebot setup is its simulation environment, which lets you test your code without a physical Turtlebot. However, seven physical Turtlebots are available in the lab, each equipped with:

- A Raspberry Pi running Ubuntu 20.04.
- A Single Board Arduino Computer (SBC) managing sensors and motors.
- A 360° LiDAR scanner which can detect objects from a distance of 16cm upto 4m away.
- An Inertial Measurement Unit (IMU) for tracking roll, pitch, and yaw.
- An odometer for tracking x and y movement.
- Stepper motors for left and right wheel control.

In this demo, We will look at some sensor output of a Turtlebot, then create a basic statechart to control a virtual Turtlebot in the Gazebo simulator, and finally operate a real Turtlebot using the same statechart model.

Quick Turtlebot Hardware Demo

This section demonstrates some of the Turtlebot sensor outputs. The command prompt format indicates whether the command is being executed on the host laptop (`user@laptop:~$`), the Turtlebot (`ubuntu@ubuntu:~$`), or a virtual machine (VM) (`user@vm:~$`). This section assumes that ROS2 (`foxy` or `humble`) and the Turtlebot packages are installed on the laptop.

1. **Connect** the laptop to the lab Wi-Fi network, `TP-LINK_8D52` (Wi-Fi password: `20406301`).
2. **Power On** the Turtlebot by connecting its battery and using the switch on the front of SBC board (boot-up takes ~30 seconds).

Turtlebots in the lab have fixed IP addresses:

- Turtlebot 1: 192.168.0.101
- Turtlebot 3: 192.168.0.103
- Turtlebot 4: 192.168.0.104
- Turtlebot 5: 192.168.0.105

- Turtlebot 6: 192.168.0.106
- Turtlebot 7: 192.168.0.107
- Turtlebot 8: 192.168.0.108

To connect, use the SSH command from the laptop with `ubuntu` as the user (password: `essess`):

```
user@laptop:~$ ssh ubuntu@192.168.0.101
```

Each Turtlebot runs on a unique ROS domain to avoid interference; this domain is controlled by the `ROS_DOMAIN_ID` environment variable. For instance, Turtlebot 1 uses ROS domain ID 1. Check the domain ID on a Turtlebot by running:

```
ubuntu@ubuntu:~$ env | grep ROS
ROS_DOMAIN_ID=1
```

To display sensor data on your laptop, set the laptop's ROS domain to match the Turtlebot's (e.g., ROS_DOMAIN_ID=1):

```
user@laptop:~$ export ROS_DOMAIN_ID=1
```

Then, launch a ROS node on the Turtlebot, which exposes sensor data and motor controls within the ROS domain:

```
ubuntu@ubuntu:~$ ros2 launch turtlebot3_bringup robot.launch.py
```

The sensor data and controls are exposed as ROS topics that we can list with:

```
user@laptop:~$ ros2 topic list
```

You can also access real-time sensor data via ROS topics. For example:

- Check battery status:

  ```
  user@laptop:~$ ros2 topic echo /battery_state
  ```

- View LiDAR data:

  ```
  user@laptop:~$ ros2 topic echo /scan
  ```

To visualize the laser scan and IMU sensor data, use:

```
user@laptop:~$ ros2 launch turtlebot3_bringup rviz2.launch.py
```

The laser data is shown as coloured dots, and the IMU data is shown as three arrows for the yaw, pitch and roll of the Turtlebot.

To control the Turtlebot from your laptop, use:

```
user@laptop:~$ ros2 run turtlebot3_teleop teleop_keyboard
```

When finished, shut down the Turtlebot properly to avoid damage by pressing `Ctrl-C` to stop the bringup command, then:

```
ubuntu@ubuntu:~$ sudo shutdown now
```

It takes around 30 seconds to shutdown. When the leds stop blinking and only two led remain lid on the Raspberry pi board, turn off the Turtlebot with the power switch on the front of the SBC board. Now the Turtlebot is fully shutdown. **Not following this procedure can damage the Turtlebot!**

Controlling a Turtlebot with a Statechart Model

Let's create a statechart model to control a virtual Turtlebot in the simulator. The figure below shows the workflow for creating and running statechart models on a turtlebot:

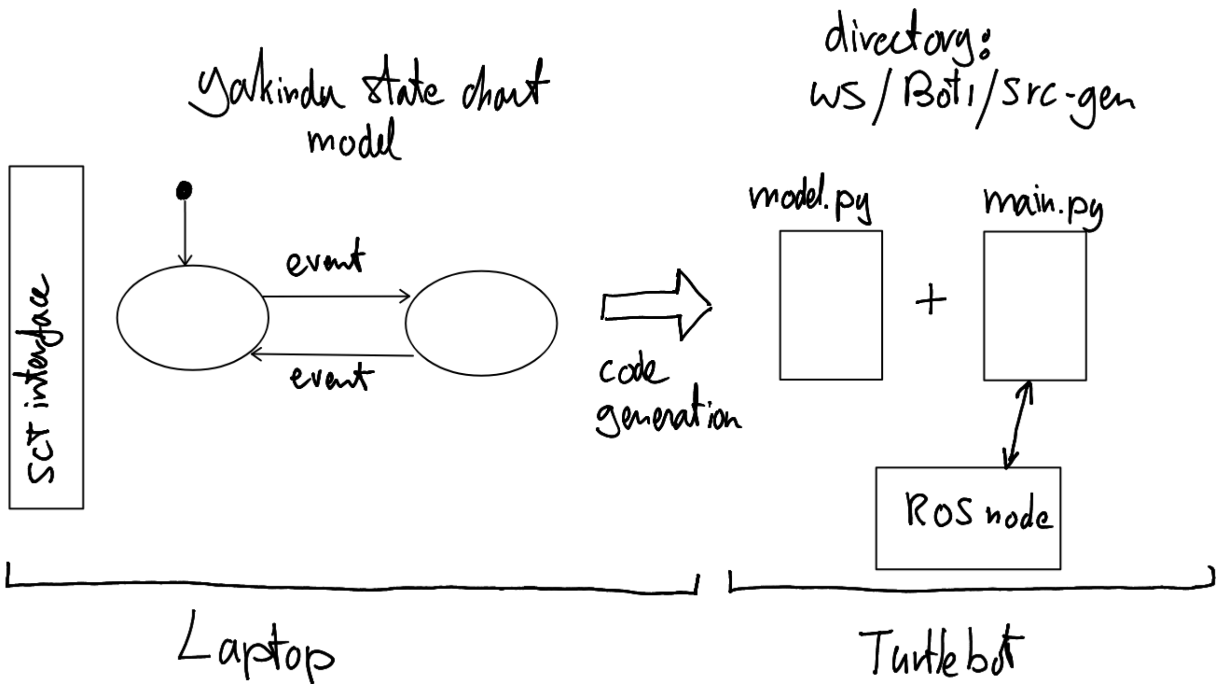

Figure: Workflow for controlling the Turtlebot with a statechart model

1. **Opening the statechart project**:

   - Open itemis CREATE in the virtual machine. You are logged in automatically as the user `ubuntu` (password: `ubuntu`). Start itemis CREATE by clicking the green itemis CREATE icon in the dock or by executing the `itemisCREATE` command in a terminal.
   - Select the workspace `itemis_CREATE/ws`. The project `Bot1` that includes the statechart file `Statechart.ysc` is already open.
   - This project contains the SCT interface used to control the Turtlebot and read sensor data. Documentation for the interface is available in `SCT_interface_manual.pdf` on Canvas.

   If you need a fresh copy of `Bot1`, download `Bot1.zip` from Canvas (or get it from the `Downloads` directory in the VM), and import it as an itemis CREATE project by navigating to **Import -> General -> 'Existing projects into workspace'** and selecting the zip archive `Bot1.zip`. Use a different workspace or delete the old `Bot1` project first to avoid duplicate projects in the same workspace.

2. **Designing a Simple Statechart Model**:

   - Create the following statechart model in itemis CREATE:

   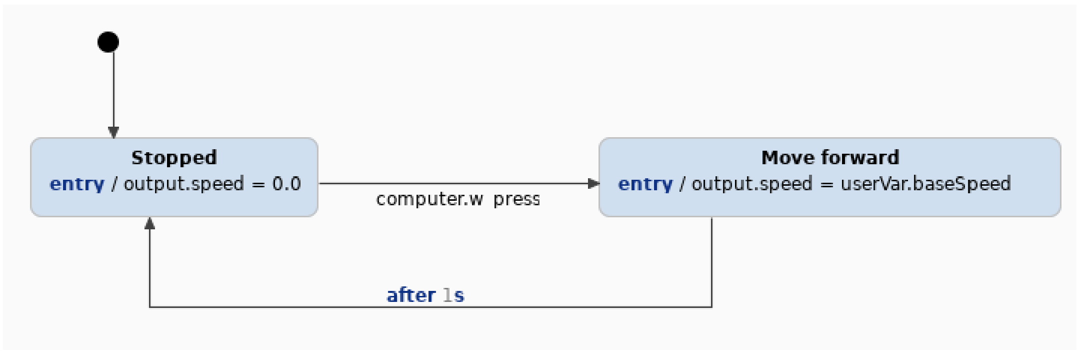
   Stopped
   entry / output.speed = 0.0
   computer.w_press
   Move forward
   entry / output.speed = userVar.baseSpeed
   after 1s

   Figure: Example of a small statechart model workflow

   - This model transitions from the `Stopped` state to the `Move forward` state when the `computer.w_press` event is triggered by pressing the "w + Enter" keys and it will return to the `Stopped` state after 1 second.

3. **Generating Python Code**:

   - When you save the statechart, itemis CREATE automatically generates a Python file, `model.py`, in the `src-gen` directory. The console message "Generating model to target project Bot1 … model done" confirms successful generation.
   - If errors are present, no new Python code is generated. Errors appear as red crosses in the statechart and also under the "Problems" tab at the bottom of the screen.

- Check if the timestamp of `model.py` has been updated because sometimes that file is not overwritten. If this happens, remove the `model.py` file, modify the statechart and save it again. You can also force Python code generation by right-clicking on `Statechart.sgen` in the project explorer window and selecting `Generate Code artifacts`.
- If the statechart editor is acting strange (eg. you cannot delete an edge or a state), close the `Statechart.ysc` statechart file and reopen it in the project explorer on the left of the screen.

4. **Running the Model on a Virtual Turtlebot**:

   - Open two terminals with the `ROS_DOMAIN_ID` environment variable set to the same value so they can exchange ROS messages. In the VM, this is set to 1.
   - In the first terminal, launch the `gazebo` simulator with:

   ```
   user@vm:~$ ros2 launch turtlebot3_gazebo empty_world.launch.py
   ```

   - In the second terminal, navigate to the `src-gen` directory and start the statechart Python code with:

   ```
   user@vm:~/src-gen$ python3 main.py
   ```

   - To move the Turtlebot forward, focus keyboard input on the second terminal and press “w + Enter”. Each press moves the Turtlebot forward for one second, exactly as we programmed in the statechart. Experiment by adding more movement commands to the statechart and testing them in the simulator. Now you are already implementing manual Turtlebot control, the first requirement of final project!

Running the Statechart Model on a Physical Turtlebot

Now let’s run the same generated Python statechart code on a real Turtlebot!

1. **Transferring the Generated Code**:

   - Copy the `src-gen` directory from the VM to your host system using a shared directory or any file transfer method. Then, transfer the directory to Turtlebot1 with:

   ```
   user@laptop:~/shared$ scp -r src-gen ubuntu@192.168.0.101:
   ```

2. **Setting Up Connections on the Turtlebot**:

   - Open two SSH shells to the Turtlebot from your laptop, connecting both to `ubuntu@192.168.0.101`.
   - In the first shell, start the ROS node with:

```
ubuntu@ubuntu:~$ ros2 launch turtlebot3_bringup robot.launch.py
```

- In the second shell, navigate to the `src-gen` directory and run the statechart Python code:

  ```
  ubuntu@ubuntu:~/src-gen$ python3 main.py
  ```

- You can now control the real Turtlebot by pressing “w + Enter” in the second shell.

If you have installed ROS locally on your laptop you can also choose to run the generated Python code on the laptop with:

```
user@laptop:~/itemisCREATE/ws/Bot1/src-gen$ python3 main.py
```

With the laptop connected to the same WiFi network as the Turtlebot and both devices sharing the same ROS_DOMAIN_ID, the Python code on the laptop can communicate with the ros node running on the Turtlebot.

Maze Exploration

To test the Turtlebot in a simulated maze with gazebo:

1. Navigate to the maze directory on the VM with:

   ```
   user@vm:~$ cd maze_world_humble
   ```

2. Generate and install a new maze using:

   ```
   user@vm:~/maze_world_humble$ ./maze_generator.py
   user@vm:~/maze_world_humble$ cp model.sdf maze_model
   ```

3. Start the simulator in the `maze_world_humble` directory:

   ```
   user@vm:~/maze_world_humble$ ros2 launch maze_world.launch.py
   ```

4. The SCT interface records the walls as the Turtlebot explores. The `src-gen` directory will contain an updated `map_generation.png` showing the maze map, viewable with your favorite image viewer:

   ```
   user@vm:~/src-gen$ sxiv map_generation.png
   ```

   For physical robots, connect to the Turtlebot over SSH with graphical forwarding (-X flag) to display the map on your laptop:

   ```
   user@laptop:~$ ssh -X ubuntu@192.168.0.101
   ```

Alternatively, if you have installed ROS locally on your laptop you can download the maze tar file from Canvas. If you installed ROS `humble`, download `maze_world_humble.tgz` and install the virtual maze exploration setup with:

```
user@vm:~$ tar xf maze_world_humble.tgz
user@vm:~$ cd maze_world_humble
```

itemis CREATE License Server

itemis CREATE requires a valid license, provided by a license server accessible through the **Window -> Preferences -> Yakindu licenses** settings. Enter `pcs-lm-01.lab.uvalight.net` as the server address, and select both licenses. You may use the "Borrow" option to work offline for up to three days. You might need to click the `retry` button to get a valid license.

You can also install itemis CREATE directly on your laptop for improved performance. You can find the download link in the TurtleBot section on Canvas.