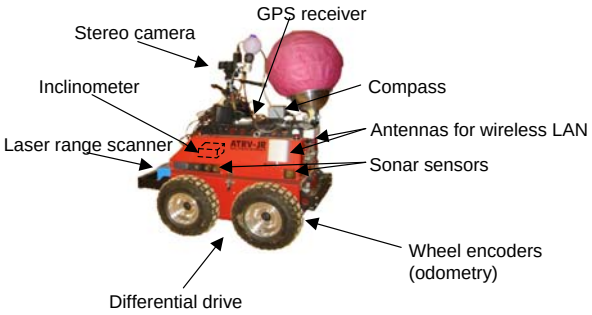
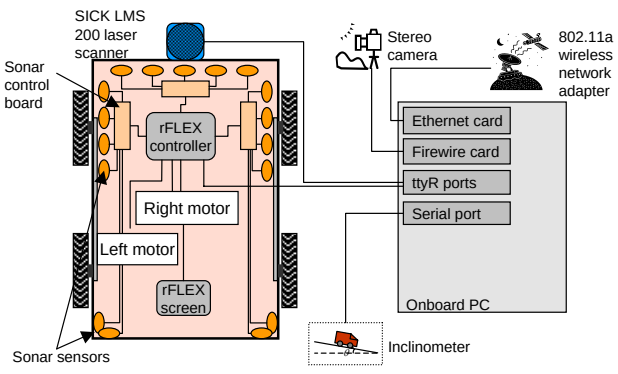
Model-Based Diagnosis and Execution on Rovers

Lars Blackmore
Steve Block
Emily Fox
Thomas Léauté

Rover Setup



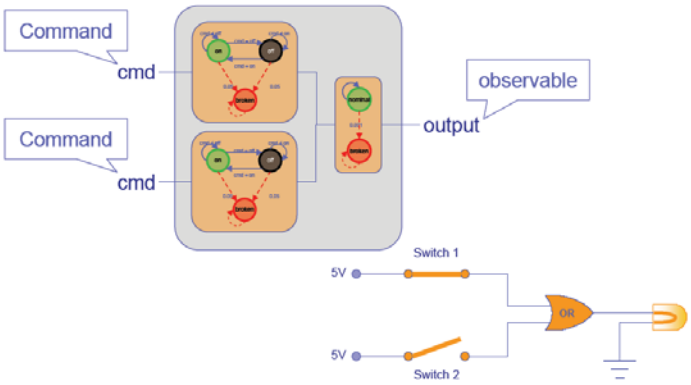
- Sensors give information on motion and environment.
- Onboard PC allows for real-time computation and command processing.



Figures from Seung Chung's Project Description handout

Titan

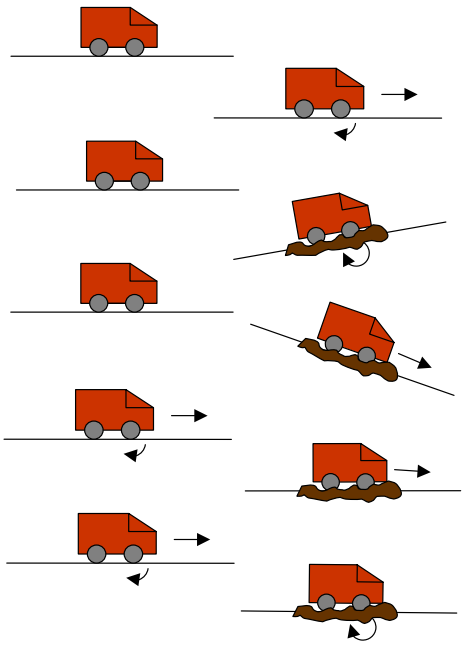
- Titan is a model-based executive deductive controller
  - Mode Estimation
  - Mode Reconfiguration
- Some components are “actuator stages” that take commands *from* Titan
- Other components are “sensor stages” that return observations *to* Titan
- Components are linked to form the complete system



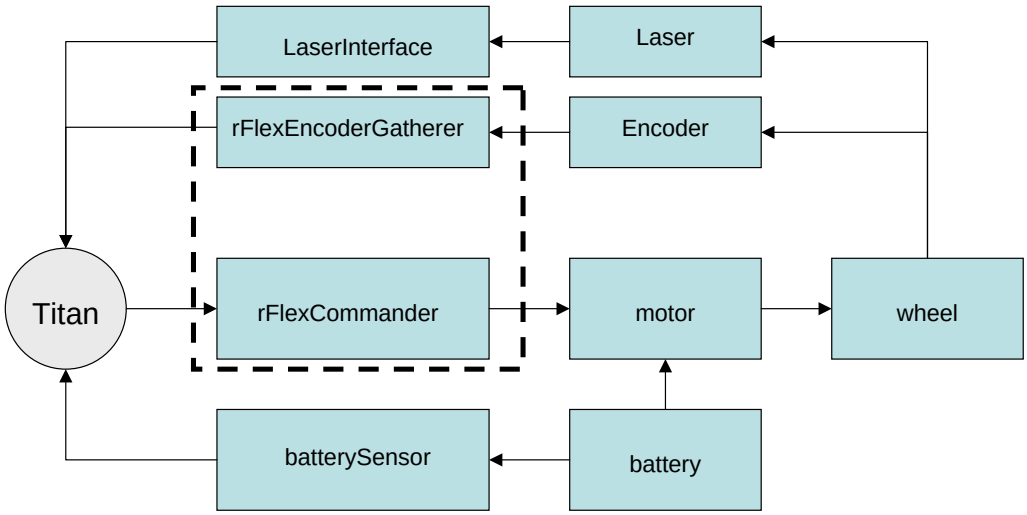
Figures from Seung Chung and Oliver Martin’s Titan Tutorial

Simple Slip-Slide Scenario

- Initialize in all-stop state
- Command ‘go’ : successful driving
- Command ‘stop’ : successful stopping
- Command ‘go’ : slips
- Command ‘stop’ : successful stopping
- No command : starts sliding
- Command ‘go’ : successful driving
- Command ‘stop’ : skids
- Command ‘go’ : successful driving
- No command : slips

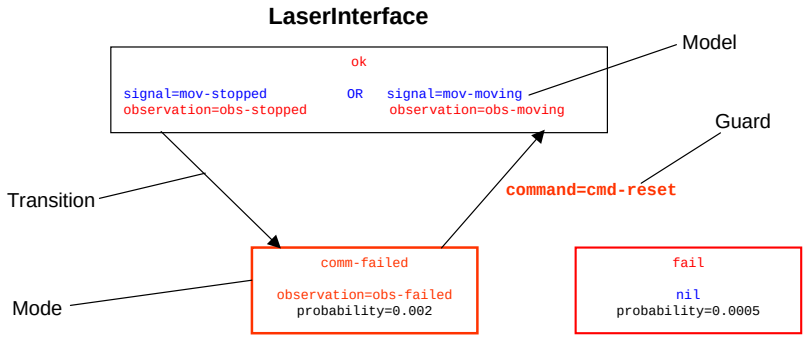


Model Schematic



Model Implementation

- Modeling is based around transition diagrams



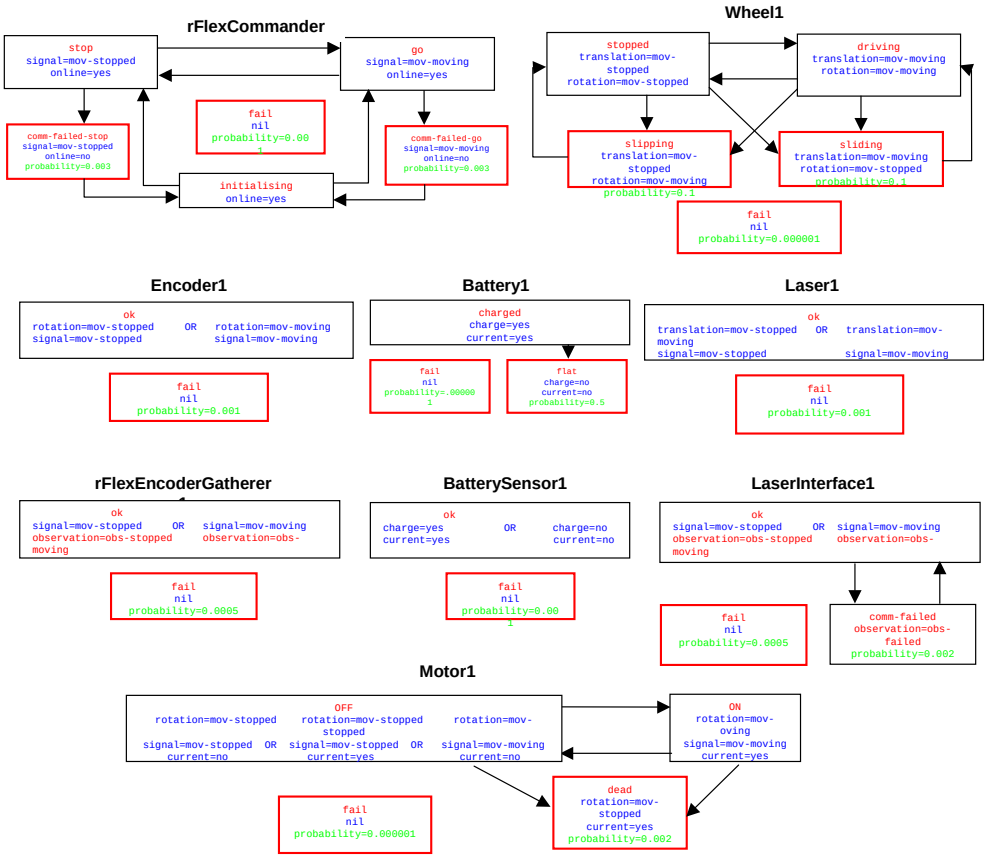
- Components are linked through relationships between the models defining the component modes

Laser1.signal = LaserInterface1.signal

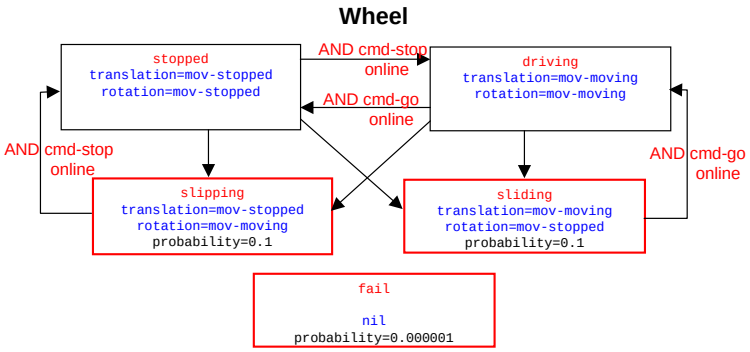
- The models of the ‘ok’ modes describe the correct operation of each component

Model Implementation

- Guards on ALL transitions to nominal states
  - Necessary for Mode Reconfiguration
- Avoid use of model variables as transition guards in other components
  - Insufficient for correct Mode Estimation
- Titan cannot handle disjunctive transition guards
  - Creative modeling required
- Components may have multiple fault modes
  - Some have defined models
  - Always one “fail” mode to handle exceptions

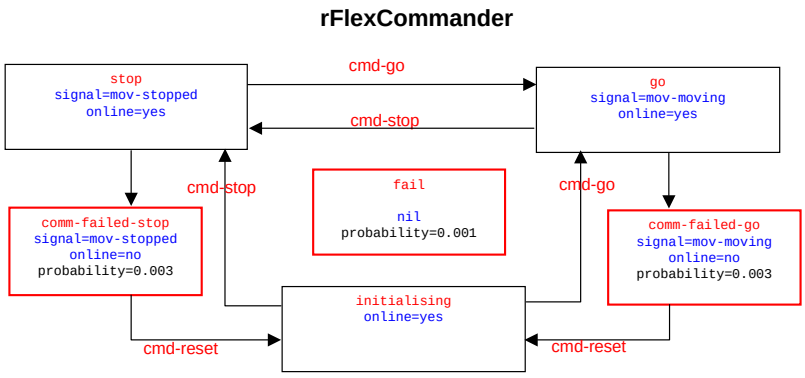


Wheel



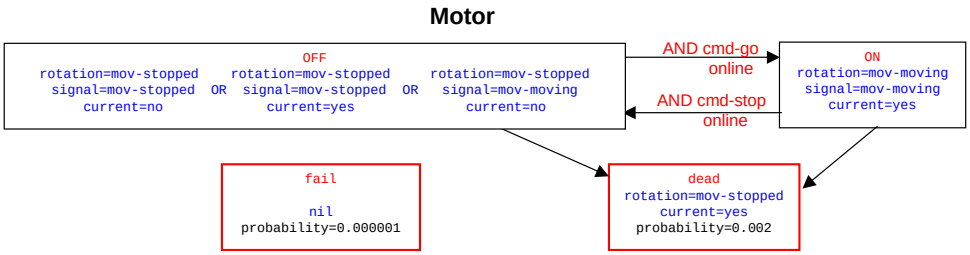
- A complete system would include separate models for each wheel
- Simplified system considers only one wheel
  - Motion of rover and wheel are equivalent
- Direction of motion not considered
- Objective is to control the state of the wheel and hence the rover

rFlexCommander



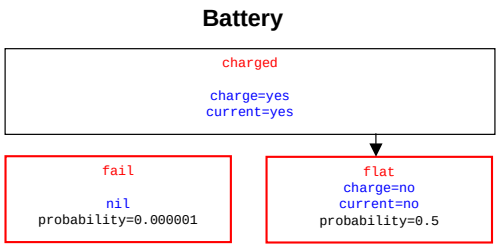
- rFlexCommander component models the portion of the rFlex board that transmits the commands from the PC to the motor.
- The “comm-failed” modes are resetable.

Motor



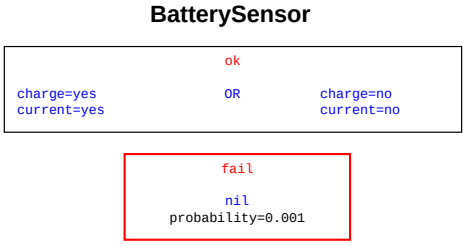
- The motor component represents the actual motor
- Takes commands from the rFlex board.
- The motor component has a “dead” state representing the failure mode when the motor is stopped and will not respond to a “start moving” command from the rFlex board.

Battery



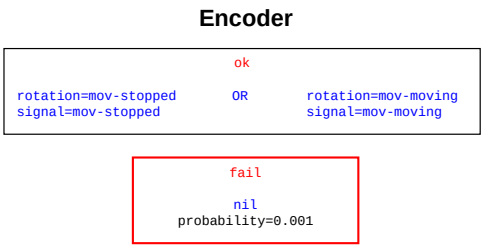
- The battery component models the power source of the rover.
- The battery component makes the overall system more realistic and adds another condition on the state of the motor.

BatterySensor



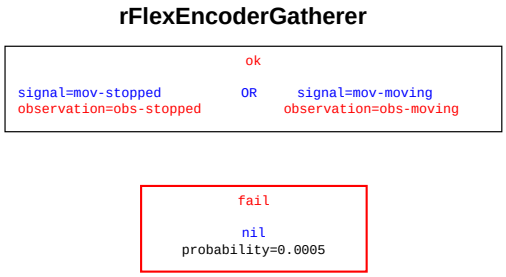
- The BatterySensor component models the way in which the system reads charge on the battery.
- The output of this component is an observable to Titan allowing it to deduce the state of the battery and motor when combined with the observables from the rFlexEncoderGatherer and LaserInterface observations.

Encoder



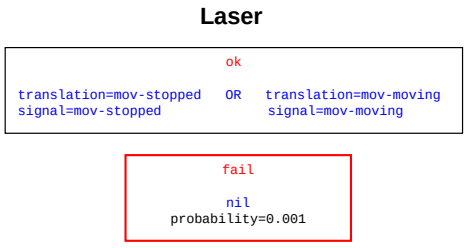
- The encoder models the shaft encoders on the wheels.
- Outputs are not directly observed because they are processed through the rFlex board.

rFlexEncoderGatherer



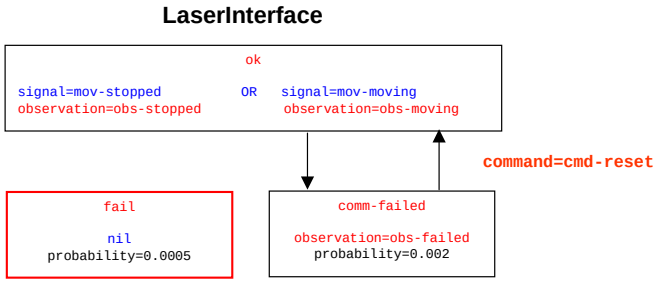
- The rFlexEncoderGatherer component models the portion of the rFlex board that receives the observations from the encoder.
- The output of this component is the actual observable in Titan.

Laser



- The laser models the laser unit on the front of the rover.
- The laser measures the rover's translation.
- Readings of the laser indicate whether the rover is moving or is stopped.
- The laser has its own interface component which is independent of the rFlex board.

LaserInterface

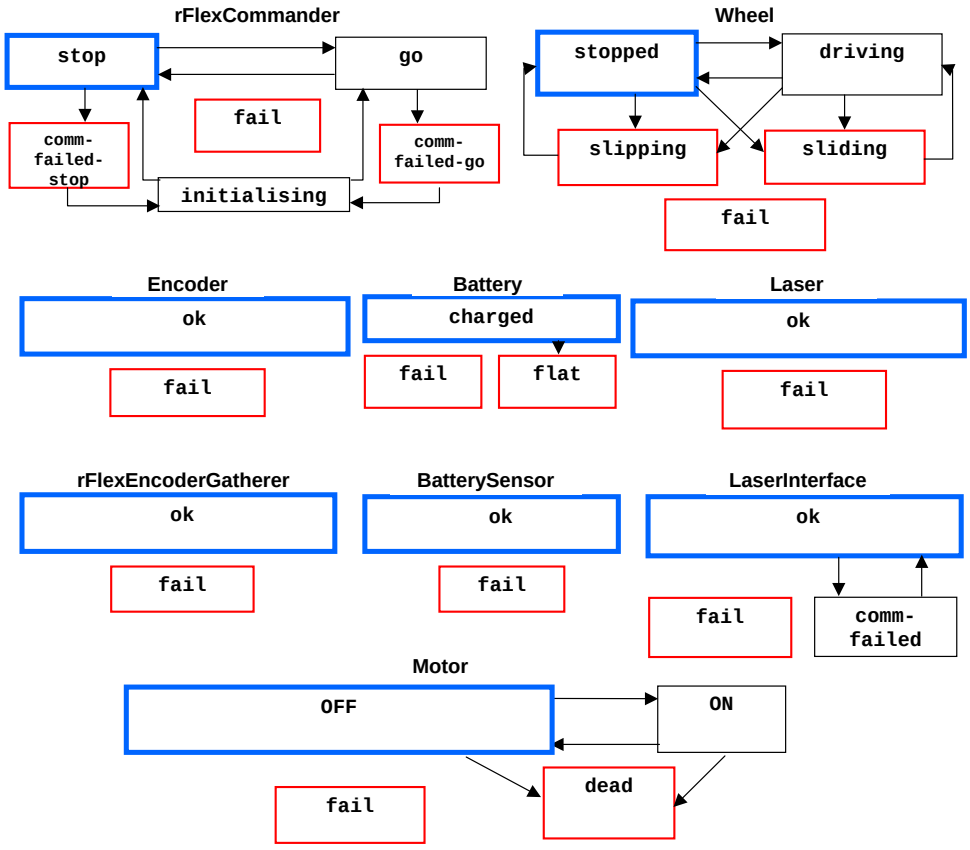


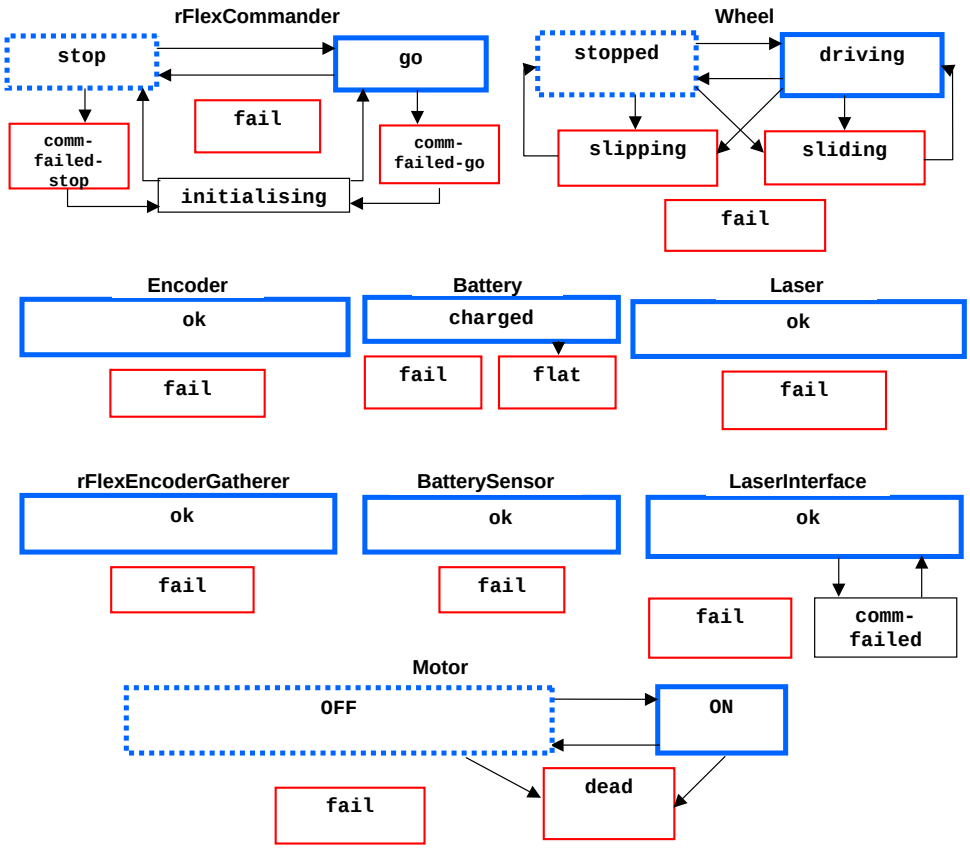
- The LaserInterface component models the interface between the laser and the onboard PC.
- The output of this component, along with the output of the rFlexEncoderGatherer and BatterySensor components, are the actual observables in Titan.

Facts

```
FACTS:
        wheel1.rotation      =     motor1.rotation
        wheel1.rotation      =     encoder1.rotation
        wheel1.translation   =     sonar1.translation

        laser1.signal        =     LaserInterface1. signal
        encoder1.signal      =     rFlexEncoderGatherer1. signal
        motor1.signal        =     rFlexCommander1.signal

        motor1.command       =     rFlexCommander1.command
        wheel1.command       =     motor1.command

        rFlexCommander1.online=    motor1.online
        motor1.online          =   wheel1.online

        motor1.current         =   battery1.current
        battery1.charge        =   batterySensor1.charge
```

The facts define relationships between the various component models.


rFlexCommander
stop
go
fail
comm-failed-stop
comm-failed-go
initialising
Wheel
stopped
driving
slipping
sliding
fail
Encoder
ok
fail
Battery
charged
fail
flat
Laser
ok
fail
rFlexEncoderGatherer
ok
fail
BatterySensor
ok
fail
LaserInterface
ok
fail
comm-failed
Motor
OFF
ON
fail
dead


rFlexCommander
stop
go
fail
comm-failed-stop
comm-failed-go
initialising
Wheel
stopped
driving
slipping
sliding
fail
Encoder
ok
fail
Battery
charged
fail
flat
Laser
ok
fail
rFlexEncoderGatherer
ok
fail
BatterySensor
ok
fail
LaserInterface
ok
fail
comm-failed
Motor
OFF
ON
fail
dead

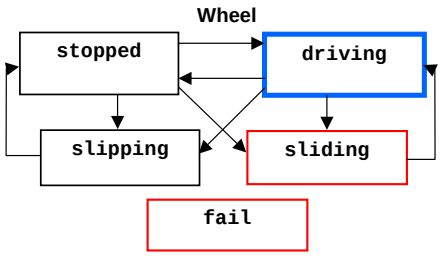
Diagnosis and Recovery Scenarios

- Slipping
- Sliding
- Laser hardware failure
- rFlex communication failure
- Laser interface communication failure
- Battery failure

Slipping Scenario

Observations:

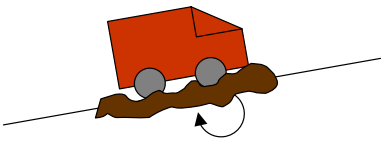
- Wheel encoder indicates rotation.
- Laser indicates no translation.
- Battery sensor indicates sufficient charge.

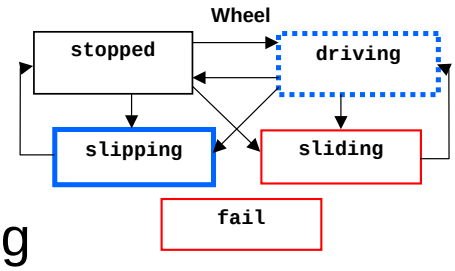


Slipping Scenario

Observations:

- Wheel encoder indicates rotation.
- Battery sensor indicates sufficient charge.
- Laser indicates no translation.



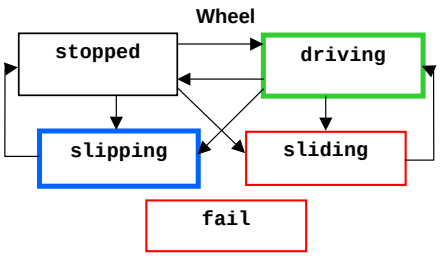


Diagnosis: rover slipping

Recovery From Slipping

Goal: rover driving

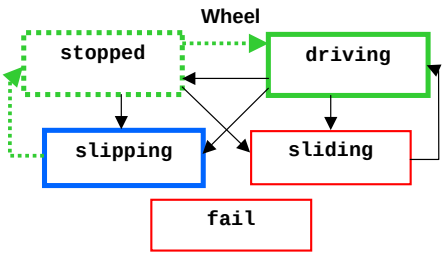
1) Find path to goal



Recovery From Slipping

Goal: rover driving

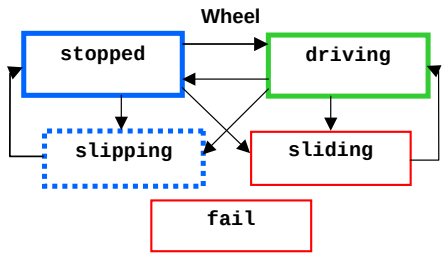
1) Find path to goal
2) Suggest command



Recovery From Slipping

Goal: rover driving

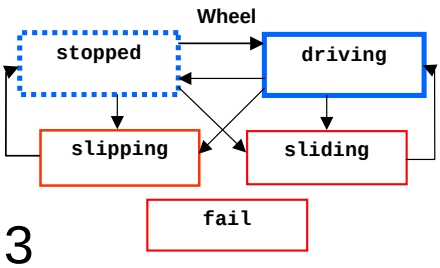
1) Find path to goal
2) Suggest command
3) Execute command



Recovery From Slipping

Goal: rover driving

1) Find path to goal
2) Suggest command
3) Execute command
4) Repeat steps 2 and 3 until goal is reached



Demonstration on Rover Testbed

VIDEO

QuickTime™ and a
DV - PAL decompressor
are needed to see this picture.

Laser Hardware Failure Scenario

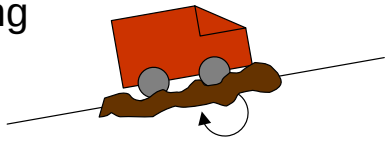
Command:

- Command wheel to start driving.

Observations:

- Wheel encoder indicates rotation.
- Battery sensor indicates sufficient charge.
- Laser indicates no translation.

Initial Diagnosis: rover slipping



Laser Hardware Failure Scenario

- Mode Reconfiguration tries to recover from slipping state by stopping wheel and driving again.
  - Laser still indicates no motion
- After three attempts, a new diagnosis becomes more probable:
  - Laser was in ‘fail’ mode from the beginning
- Mode Estimation determines that wheel is in ‘driving’ mode as required
  - No reconfiguration necessary

rFlex Communication Failure Scenario

- Very similar to laser hardware failure.
- Titan makes several attempts at resetting the commander.
- Eventually deduces that motor has been burnt out from start.

Demonstration on Rover Testbed

VIDEO?? QuickTime™ and a DV - PAL decompressor are needed to see this picture.

Full Slip-Slide Scenario

- Tested in Titan providing observations and commands manually
- Successfully estimates state of rover
- Suggests correct recovery trajectory
- Implemented on rover
  - Physical situation can however not be simulated

Other Scenarios

- Diagnosis is simple
  - Sensors can detect failure directly
- Reconfiguration is either simple or impossible
  - Reset LaserInterface
  - Charge battery (cannot be done by PC)
  - Burnt-out motor (irrecoverable)
- Off-line testing in Titan successful
- Demonstration on rovers in progress
  - Necessary interfaces in place
  - Update of onboard Titan code required

Conclusion

- Successfully developed a model of ATRV rovers in Titan
  - Incorporated complex features
  - Touched boundaries of current Titan capabilities
- Developed test scenarios
- Tested feasible scenarios on rover testbed
  - Incorporated sensor interface software

Questions