

Model-Based IoT Translator

Sung-Jun Baek, Cael Shoop, Cameron Wright

Faculty Advisor: Dr. Siddhartha Bhattacharyya, Dept. of Computing and Engineering,
Florida Institute of Technology

Problem Statement

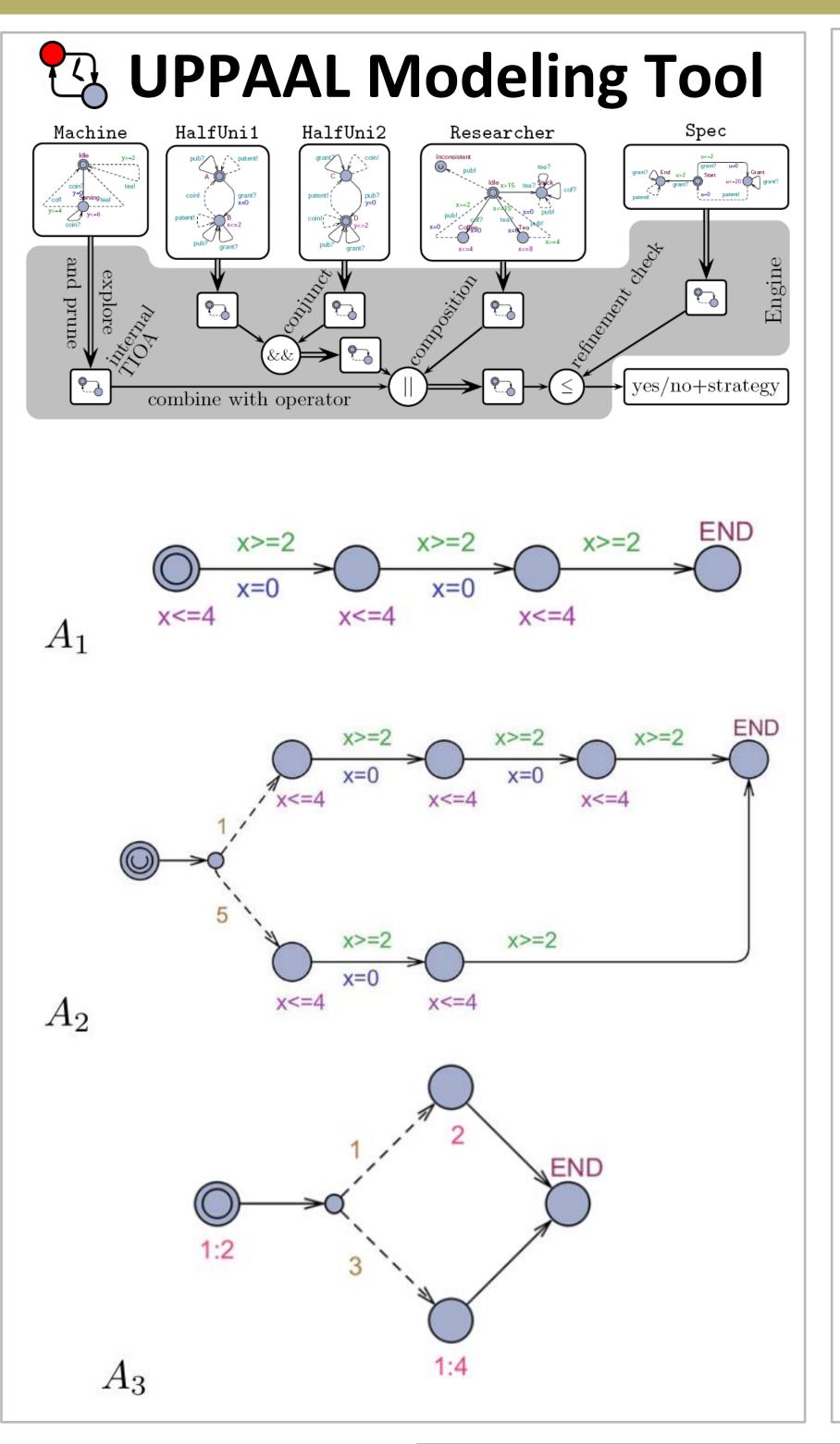
- Upon IoT system design, developers have to hard-code for every device on architectural logic change
- Uneasy to add devices to existing inter-operational IoT systems

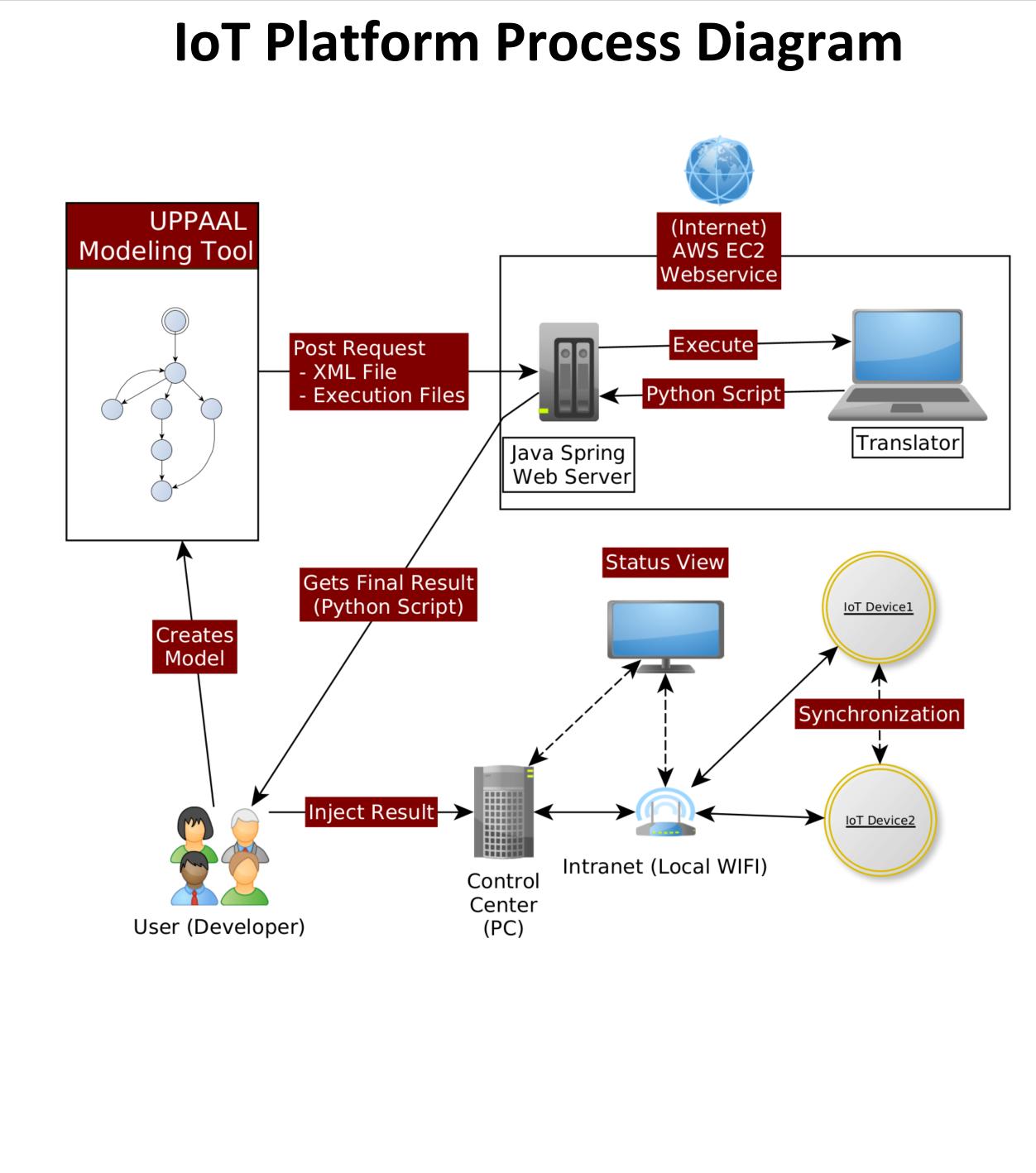
Primary Project Goal

- Build a software program that does the distributed autonomous system for mission excavation
- Convert modeling diagrams into codes that can be applied to machines
- Able to perform a visual presentation by running codes on actual devices for the demonstration

Current Translator Features

- Translator can accept and generate codes based on UPPAAL models
- Translator can accept predefined device execution files
- Website application is provided for increased accessibility of the tool
- Translator can accept multiple synchronized model templates





Conclusion

Model-based IoT Translator is a logical software that translates the UPPAAL model into entity objects to produce executable and deployable codes for IoT devices.

Future Direction

- More features on inline code handling
- Interoperability with other programming languages
- Increase utilization of predefined codes
- Simulator for the Translator output

Potential Application

- Home appliance control infrastructure
- Planet exploration infrastructure
- Security IoT Platform
- Car IoT platform
- IoT devices using Python script

Visual demonstration

Translator webservice

class MCCD:

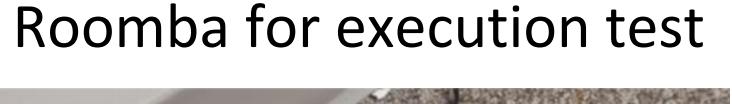
async def close(self):

await self.close()

await self.wait_process()

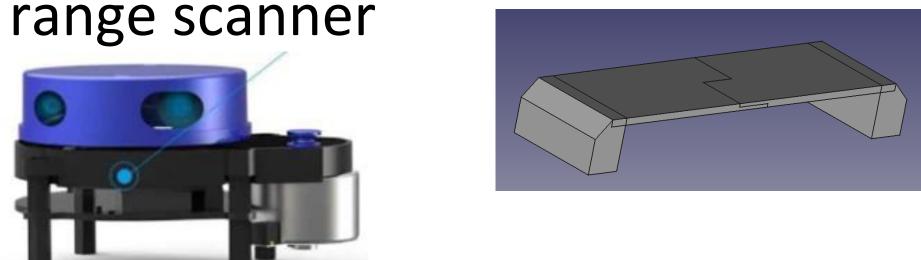
async def Initialize(self):

def __init__(self,):
 pass

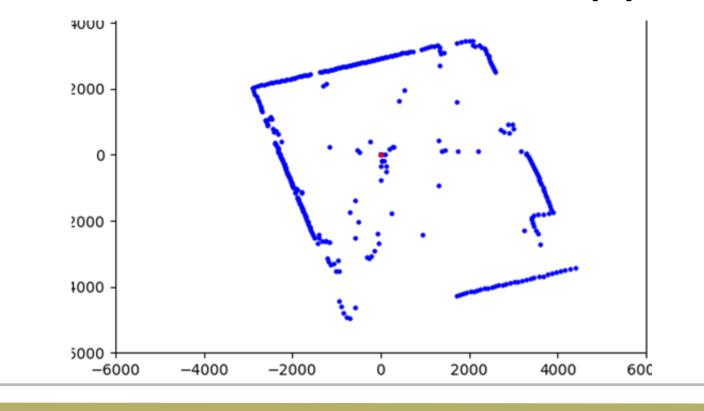




Rotational laser 3D Printed Mount



Visual Result of Room Mapping



main(arguments) XML_parser.py tree: xml.etreeElementTree global set: GlobalSet init (xml file name) generate model(xml file Sends Data parse predefined files(files) parameter_parser.py transition_parser.py TransitionParser get_result_data(PredefClass function_obj.py PredefGlobalObject PredefFunction global_obj.py import_obj.py PredefImport variable obj.py Transition translator Node class_gen.py ClassScriptGen -GlobalSet -PredefGlobalObject function_gen.py Variable FunctionScriptGen py_export.py Export

script_gen.py

TranslateModel

Combined and generated

System Process and Architectural Diagram of Model Translator