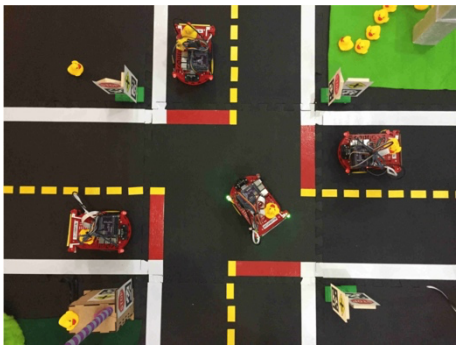
CS486C – Senior Capstone Design in Computer Science

Project Description

Project Title: Visual Programming for Duckietown Autonomous Robots	
Sponsor Information: Intelligent Control Systems (ICONS) Laboratory	**Truong X. Nghiem**, Assistant Professor School of Informatics, Computing and Cyber Systems Northern Arizona University Email: truong.nghiem@nau.edu Phone: 928-523-4973

Project Overview:



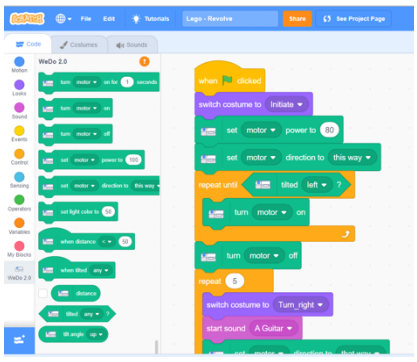
Waymo (Google), Uber, Tesla, Amazon, and hundreds of startups all over the world are putting huge efforts into building autonomous vehicles, including self-driving cars on the road and autonomous robots delivering food and products.

Many hardware and software platforms have been developed for research and education in autonomous robots, particularly self-driving cars. For example, the F1tenth platform of autonomous race cars (https://f1tenth.org/) and the Duckietown platform of small autonomous robots in a miniature town (https://www.duckietown.org/). Both platforms are available and being actively used for research and education in the Intelligent Control Systems (ICONS) Laboratory in SICCS.

The problem. The Duckietown platform in the ICONS Lab, in addition to its value as a research platform, serves as a motivating and powerful educational platform as well, allowing students to use them to explore the basics of robotics and autonomous vehicles. However, programming the autonomous Duckie robots requires good to excellent skills in a programming language such as Python or C++, considerable technical expertise in embedded systems, and a good understanding of general robotics, computer vision, control, planning, and even machine learning/AI. These requirements pose significant challenges and set a high barrier to entry, making them suitable as educational platforms only for upper division courses in university CS programs; though potentially great for engaging high school and middle school students in the exciting autonomous vehicles area, Duckietown is currently inaccessible to these audiences, especially of underrepresented groups in STEM.

The Problem. The goal of this project is to investigate the feasibility of developing a relatively simple visual programming tool that, while almost certainly more limited in capabilities, can make fun, basic exploration of autonomous vehicle programming in Duckietown more accessible, i.e., will make it fairly simple and intuitive for high-school and middle-school students with only moderate technical skills to program the Duckie robots in some basic exploratory ways, allowing them exciting exposure to the basics of robotics, control and autonomous driving technologies. Such a tool will lower the barrier for high school and middle school students to enter and prepare them for a future career in this STEM field.

Envisioned Solution



The overall goal of this project is to develop a Visual Programming Tool (VPT), or more likely an extension to an existing VPT framework, for programming the Duckietown robots. The idea is inspired by projects and tools such as Scratch Link and LEGO BOOST (https://scratch.mit.edu/boost) for programming LEGO robots with Scratch, and LabVIEW from National Instruments. The tool will provide building blocks for interfacing with a Duckietown robot and for commonly used functions (such as object detection, motor control, basic path planning), and a visual programming environment to connect the blocks to construct a working autonomous robot program. The effort can be split into two main tasks:

Task 1 (feasibility and design): Investigate the product feasibility constraints and develop a product design

The team will explore different options for developing a VPT for Duckietown, write a feasibility assessment report, and develop a design of the final product. Duckietown is an open-source project, therefore all software code and hardware designs are publicly available. This task can be divided into the following subtasks:

1. The team learns about the Duckietown platform, their programming, and general robotics (such as object detection, planning), using the platform available in the ICONS Lab and a multitude of excellent free tutorials and online courses (edX's "Self-Driving Cars with Duckietown" course).
2. Survey different existing, preferably open-source, visual programming languages/tools (VPTs) and explore the feasibility of basing the final product on each option (e.g., what does it entail to extend an existing tool for Duckietown? what is the long-term prospect of the existing tool?). Some examples are: Scratch (open-source), NEPO/Open Roberta (open-source), LabVIEW (proprietary), Simulink (proprietary). The team then writes a feasibility assessment report of the investigated options.
3. Combining the outcomes of subtasks 1 and 2, select the most viable option and develop a design of the final product. The design must also include what building blocks or features related to the Duckietown platform (i.e., the *Duckietown library* for the VPT) are going to be provided by the product, for a *minimal usable product* and a *truly complete product* (which has a more complete *Duckietown library)*. The determination of minimal usability and true completeness will be discussed with the sponsor.

It is expected that Task 1 will be completed by the end of the Fall semester.

Task 2 (product development): Develop a working prototype of the Visual Programming Tool for Duckietown

Using the design developed (and the robotics and Duckietown knowledge and skills learned) in Task 1, develop the Visual Programming Tool for Duckietown. The development includes at least two main parts:

1) The interface between the selected VPT and the Duckietown platform, e.g., the communication between them, the way a program written in the selected visual programming language is built and downloaded to the robots, the real-time interface between the tool and the running robots.
2) The *Duckietown library* of building blocks and features for the Duckietown platform in the chosen VPT, e.g., a block for detecting objects using the robot's camera, a block for changing the speed of the robot, a block for changing the direction (turning) of the robot, etc.

A *minimal usable product* with a *basic Duckietown library* covering the most common and basic usage of the robots is expected at the end of this task. A *truly complete product* with a richer / more complete *Duckietown library* is desirable, and client satisfaction will be commensurate with completeness in this direction.

Project Impact

A summer camp for high school or middle school students may be held in Summer 2022 for exploring robotics and self-driving cars. Ideally, the students will use the software developed in this Capstone project and the Duckietown platform in ICONS Lab to program their own autonomous robots. Such activities and experiences will help encourage them to pursue STEM in the future.

Knowledge, skills, and expertise required for this project:

- Must have software development skills and programming skills in Python and/or C/C++.
- Familiarity with Linux and building programs from source code in Linux.
- Knowledge and experience of Computer Vision, Embedded Programming, Control, Robotics, Robot Operating System (ROS), and a visual programming language/tool (e.g., Scratch) is an advantage.

Equipment Requirements:

- The sponsor will provide the Duckietown platform (hardware & software).
- Students must have personal computers on which Ubuntu Linux can be installed (either directly, on a VM, or on Docker).
- Otherwise, there should be no equipment or software required other than a development platform and software/tools freely available online.

Software and other Deliverables:

- The software product as described above, deployed and tested successfully with the Duckietown platform. Must include a complete and clear User Manual for configuring and operating the software.

- A strong as-built report detailing the design and implementation of the product in a complete, clear and professional manner. This document should provide a strong basis for future development of the product.
- Complete professionally-documented codebase, delivered both as a repository in GitHub, BitBucket, or some other version control repository; and as a physical archive on a USB drive.